1972

A PL360-based compiler generating system.

Woods, Robert Allen.

Monterey, California. Naval Postgraduate School

A PL360-BASED COMPILER GENERATING SYSTEM

Robert Allen Woods

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A PL360-BASED COMPILER GENERATING SYSTEM

by

Robert Allen Woods

Thesis Advisor: R. H. Brubaker

December 1972

Approved for public release; distribution unlimited.

A PL360-Based Compiler Generating System

by

Robert Allen Woods
Lieutenant, United States Navy
B.S., Kansas State University, 1965

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1972

ABSTRACT

A compiler generating system written in the language PL360 to run on IBM System/360 computers is presented. The concepts and principles of the XPL compiler generating system are reviewed. The SLR(k) parsing algorithm is briefly described, and an example of SLR(1) parsing is presented. A description of the compiler generating system is presented along with its limitations, and instructions for its use are given. The required PL360 to System/360 interface is described and a listing is included. Program listings and sample input and output are included in the appendices.

TABLE OF CONTENTS

I. INTRODUCTION

Most computer programs are not written in a machine language. Therefore, it is necessary that a program (assembler, compiler) first translate the input sentences of the source program into an appropriate sequence of machine instructions before execution can begin. The task of writing this program, or translator, is often a long and tedious process. Computer Science research and experience with existing translators have made it possible for researchers to automate major portions of the task.

Most translators have many tasks in common, such as scanning text, analyzing syntax, synthesizing code, and interacting with an operating system. It is these tasks that are automated by translator writing systems (TWSs). Once these tasks are handled, the compiler writer can concentrate on those items unique to his particular translator.

The objective of the research reported herein was to complete the development of a TWS based upon the language PL360 [Ref. 1] and to implement the completed TWS at the W. R. Church Computer Center, Naval Postgraduate School. This goal has been achieved by completion of the proto-compiler (a syntax checker without code synthesis facilities, a prototype compiler) originally written by Blanchard [Ref. 2] and the development of a PL360 program to analyze a grammar and produce the corresponding tables required by the SLR(1) [Ref. 3] parsing algorithm.

The next chapter of this report contains a review of one TWS and a description of the SLR(k) parsing algorithm. A brief description of the language PL360 is also presented to provide the reader with some background information. Chapter III contains a description of the completed PL360 compiler generating system. The required OS/360 interface is presented in Chapter IV.

II. BACKGROUND

This chapter first explores the concepts and principles of one TWS and then presents a review of the parsing algorithm used for the PL360-based TWS. The translator writing system reviewed is the XPL system of McKeeman, et al. [Ref. 4]. The parsing algorithm, SLR(k), is that chosen by Blanchard as the basis for the proto-compiler. An excellent paper by Feldman and Gries [Ref. 5] contains a critical review of many TWS efforts. Section C contains background information concerning the language PL360.

A. THE XPL SYSTEM

In this section, the principles of McKeeman's compiler generator are discussed. The system is explained in detail in Ref. 4, which serves both as an introduction to the construction of TWSs and as a user's manual for the XPL programming language.

The parsing algorithm used by McKeeman is the mixed-strategy (MSP) algorithm, a particular type of bottom-up parser, which is a modification of Wirth's [Ref. 6] precedence concept. The distinguishing feature of the algorithm is that it does not use state-of-the-parse information, as top-down methods do; rather, it involves examining the canonical sentential form (each string in a canonical parse) to determine what unique parse step is applicable and then performs a substitution.

The three major programs of the XPL compiler generating system are the syntax analyzer which builds the tables required by the MSP algorithm, the proto-compiler with which the user can produce a compiler, and the XPL compiler which translates XPL statements to System/360 machine code.

The syntax analyzer is a program which accepts the BNF definition of a grammar, determines whether the grammer is MSP(2,1:1,1) (a special case of MSP), constructs parsing decision tables, and punches those tables on cards in the form of XPL declarations.

The proto-compiler uses the cards produced by the analyzer and functions as a syntax checker. The user may build on the proto-compiler by rewriting the code synthesis routine to implement the semantics of the new language to be compiled and by altering the text-scanning routine to process the terminal symbols of the new language. During syntax analysis each reduction causes the code synthesis procedure to be invoked, and the appropriate machine code can be generated.

The XPL compiler generating system allows the user to construct compilers for languages described by grammars with a minimum of effort.

B. THE SLR(k) PARSING ALGORITHM

DeRemer defines a class of context-free grammars called "Simple LR(k)" or SLR(k) which includes the simple precedence grammars as proper subsets. A method for constructing parsers for SLR(k) grammars is given in Ref. 9. SLR(k) parsers have been implemented by DeRemer and appear to be

superior to corresponding MSP parsers both in the speed of parser construction and in the size and speed of the resulting parsers.

For the stacking decision, the MSP algorithm uses at most the top two symbols on the parse stack and the next symbol from the input text. SLR(k) parsers make the stacking decision based on all the symbols in the parse stack plus k more from the input text. This is accomplished by restructuring the stack and saving state-of-the-parse information. The operation of the parser will be illustrated by example.

Consider a sample grammar:

G :: = E

E :: = E + T | T

T : : = (E) | x

The finite state machine represented in Fig. 1 parses sentences generated by the sample grammar. The algorithm is started in state 0 and passes through a series of states until reaching a state with no successor. The indicated rule is applied and the parser is restarted in state 0. The algorithm terminates upon reaching state 2 and encountering an end-of-file mark. For example, when in state 1 and the symbol "x" is encountered, apply the reduction T x and restart; when in state 3 and the symbol "(" is encountered, stack the symbol and enter state 4.

The SLR(1) parsing algorithm has been implemented in Blanchards proto-compiler.

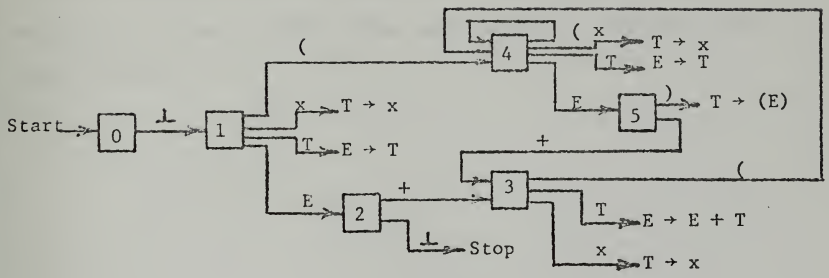

Fig. 1 Finite State Machine for the Sample Grammar

C. PL360

In 1968, Wirth published a formal description of PL360 [Ref. 1], a language designed specifically for the IBM System/360. A year later, a compiler for PL360 was written by Wirth, J. W. Wells, Jr., and E. Satterthwaite, Jr. and made available through the IBM Contributed Program Library [Ref. 7]. Several amendments to the original language definition were included with the documentation issued with the compiler. Further extensions and modifications to the language have recently been carried out, most notably by M. A. Malcolm [Ref. 8]. Malcolm's PL360 manual has incorporated all changes to the language definition and compiler description to date.

PL360 is a language that provides most of the facilities provided by System/360 Assembler Language yet exhibits a block-structured, ALGOL-like syntax. It was designed to

improve the readability of programs which take advantage of features unique to the System/360. The PL360 language is defined by a set of BNF rules and semantic explanations given in Ref. 8.

III. DESCRIPTION OF THE PL360 COMPILER GENERATING SYSTEM

This chapter describes the completed compiler generating system that was begun by Blanchard. To complete the system it was necessary to:

1. Implement in PL360 a syntax analyzer that would provide the necessary SLR(1) parsing tables required for the proto-compiler.
2. Make several changes to the proto-compiler to make it more versatile.

A. THE SYNTAX ANALYZER

1. Description of the Syntax Analyzer

The program called SLR1ANAL is the grammar analyzer of the compiler generating system. A program listing is contained in Appendix A. It is patterned after DeRemer's Simple LR(1) grammar analyzer that was written in XPL and has the same function and basic structure.

SLR1ANAL constructs a Characteristic Finite State Machine for a grammar and produces the SLR(1) parsing tables that are required by the proto-compiler. Appendix B contains a listing of a sample input grammar. The output of the SLR(1) parsing tables is in the form of punched PL360 declarations. Appendix C contains a complete listing of the SLR1ANAL output for the input shown in Appendix B.

The program can be divided into 7 basic parts:

1. The data area which contains 9 PL360 data segments that contain the numerous arrays that are required.
2. The procedure READG that reads the input BNF description of a grammar, sorts the grammar, finds the goal symbol of the grammar and prints the grammar in standard BNF format.
3. The procedure COMPUTECFSM that computes and outputs the Characteristic Finite State Machine.
4. The procedure LOOKAHEAD that computes and outputs, if necessary, the required look-ahead sets for the SLR(1) parser.
5. The procedure COMPUTEDPDA that computes and outputs the look-back states and sets the successors of the reduce states.
6. The procedure PUNCHDPDA that prints and punches the declarations required for the proto-compiler.
7. The main program that controls the logical flow of the computation.

2. How to Use the Syntax Analyzer

Input to SLR1ANAL is made by cards containing BNF productions which are placed one to a card by using the following conventions: if the first column is non-blank, the first token is taken to be the left part of the production; otherwise, the left part is assumed to be the same as the left part of the preceding production. The balance of the

card is taken to be the right part of the production. Any token that does not occur as a left part is a terminal symbol. Any token that occurs only as a left part is taken to be the goal symbol for the grammar. All productions with the same left part must be grouped together. If the user wishes to recognize identifiers and/or numbers he should include as terminal symbols <IDENTIFIER> and <NUMBER>, respectively. The proto-compiler has mechanisms to recognize these symbols and take the necessary actions.

Input cards with the character "@" in the first column are treated as comment or control cards as listed below. It is possible to batch grammars together by separating the grammars by control cards beginning with "@EOG". Such a card should not be included after the last grammar. The SLR1ANAL control cards are as follows:

a. @I is a toggle controlling the listing of the input data cards (initially off).

b. @G is a toggle controlling the listing of the reformatted grammar (initially on).

c. @C is a toggle controlling the listing of the configuration sets (initially off).

d. @F is a toggle controlling the listing of the CFSM (initially on).

e. @L is a toggle controlling the listing of the look-ahead sets (initially on).

f. @D is a toggle controlling the listing of the DPDA (initially on).

g. @P is a toggle controlling the punching of the parsing tables (initially off).

h. @EOG separates batched grammars.

i. All other cards containing "@" in column one are taken as comment cards.

The job control language required to execute SLR1ANAL can be found in Appendix G.

3. Limitations of the Syntax Analyzer

SLR1ANAL reads a grammar and attempts to construct an SLR(1) parser for it. The program either succeeds and punches tables that represent the parser, or it prints messages stating the reasons why it cannot. If the program fails, the given grammar is either too large in some aspect or it is not SLR(1); i. e., the program will suceed, except for space limitation, for any SLR(1) grammar. The one limitation most notable is that the BNF description of an input grammar cannot exceed 250 productions.

B. THE PROTO-COMPILER

1. Description of the Proto-Compiler

The program called "PROTOCOM" forms the basis of the PL360 compiler generator system. A listing of PROTOCOM is given in Appendix D. It is patterned after McKeeman's proto-compiler and has the same function and basic structure.

PROTOCOM uses the SLR(1) parsing tables obtained from SLR1ANAL. The tables are referenced by the algorithm contained in the procedure ANALYZE to implement the finite state machine of a grammar.

ANALYZE calls on procedure PUSHANDREAD or procedure SYNTHESIZE based on a decision to stack the current symbol and read a new one or to make a reduction and perform the required semantic operations. Since code synthesis is not a function of PROTOCOM, SYNTHESIZE exists only to maintain flow of control and indicate that its presence would be required in a full-scale compiler.

Procedure SCAN, called from either PUSHANDREAD or ANALYZE, interprets characters in the card buffer. Upon reaching the end of a card image, a call on procedure GETCARD causes a new card to be read.

The only other major procedure is ERROR which is called from a number of other routines. It handles syntax errors and prints diagnostic messages.

PROTOCOM as implemented by Blanchard would accept as terminal symbols only those consisting of one character, with the possibility of future expansion to eight characters. One other limitation was the data area available for the parser tables. The mechanisms causing these limitations were changed to allow the proto-compiler to be useful for large grammars. PROTOCOM will now recognize all symbols of 64 characters or less, and the data area available for the parsing tables is limited only by the space available in the user's System/360 region.

2. How to Use the Proto-Compiler

The first step in using PROTOCOM is to place the punched declarations from SLRlANAL into the beginning

of PROTOCOM. They must be placed after the comment "THE FOLLOWING CARDS WERE PUNCHED BY THE SLR(1) SYNTAX ANALYZER" and before the comment "END OF CARDS PUNCHED BY SLR(1) SYNTAX ANALYZER." The cards that are already present between the two comments must be removed.

Input to PROTOCOM is a program written in the language defined by the grammar. PROTOCOM will parse the input program. If it detects any conflicts between the input program and the SLR(1) parsing tables, it will output an appropriate error message along with a partial parse. If no errors are found, the only output is a listing of the input program and the message "END OF COMPILATION." A listing of a sample output is located in Appendix E.

The job control language required to execute PROTOCOM is given in Appendix G.

3. Limitations of the Proto-Compiler

PROTOCOM appears to have no limitation as a syntax checker. If the user wishes to use PROTOCOM as the basis for a compiler, he must write the procedures SYNTHESIZE and EMIT. These procedures are necessary to generate the machine code necessary to execute the program. At present PROTOCOM does not build any symbol tables. Therefore, the user must also write the procedure LOOKUP to enter and look up symbols in the symbol table.

IV. OPERATING SYSTEM INTERFACE

Since PL360 object modules do not communicate directly with an operating system, an interface program must be provided that can be compiled separately and linked to the PL360 object module by the linkage editor or linking loader.

A program called "PLIO" is presently being used as the interface between the PL360 object modules and OS/360. A listing of PLIO can be found in Appendix F. The current interface contains 4 subroutines which facilitate input and output operations. In addition to these 4 subroutines, 2 more subroutines need to be added to PLIO if PROTOCOM is to be used as a compiler. The first subroutine would decode any required parameter list, open required data sets, obtain free storage, and supply system identification. The second would release free storage and close the required data sets.

The subroutines presently in PLIO are listed below with their names and specifications.

A. READ: Read an 80 character record from the input data set and store it in the 80-byte area designated by the address contained in R0. If an end-of-file is encountered, then set the condition code to 2, else set it to 0.

B. WRITE: Write a 132 bype record from the memory location designated by the address contained in RO to the output data set.

C. PAGE: Causes the next output record to contain the USASI control character "1" to be placed into the first position of the next output record.

D. PUNCH: Write the 80-byte record whose address is contained in R0 to the punch output data set.

PLIO uses the following input and output data sets, which are identified by their DDNAMES. All data sets are sequential with fixed block format.

A. SYSIN: This data set contains the input and is referenced by the READ subroutine. The data set consist of compiler instructions and one or more source programs.

B. SYSOUT: This data set contains output originating with the subroutine WRITE and any compiler diagnostic messages. The logical record length is 133 bytes.

C. SYSPUNCH: This data set contains output originating with the subroutine PUNCH. The logical record is 80 bytes.

If PROTOCOM is used as a compiler then one data set with a logical record length of 80 bytes and closed with a disposition of REREAD would be required to contain object module output. The data set SYSPUNCH could be used for this purpose and passed to the following step (i.e., LINK).

V. CONCLUSIONS

The PL360 language is well suited to form the basis of a compiler generating system which is to be implemented on the IBM System/360. It is fairly successful in providing a tool which is superior to assembly code and in meeting the objectives of readability and writability. The language is not as easy to use as some other high-level compiler-writing languages, such as XPL. The execution speed, however, indicates that it may be the superior language in systems and production programming applications.

To make the compiler generating system useful, rigorous debugging of the system should be done. The aforementioned procedures in PROTOCOM , EMIT and LOOKUP, need to be written, along with the nucleus of the procedure SYNTHESIZE. A completely documented and explanatory users' manual is needed to make the system easier to use. It is recommended that these projects be undertaken to make the system a truly useful compiler generating system.

APPENDIX A

SYNTAX ANALYZER LISTING

```
$TITLE   SLR1ANAL                                                       0001
BEGIN                                                                   0002
                                                                        0003
    COMMENT  THE FOLLOWING GLOBAL DATA SEGMENTS CONTAIN ALL OF          0004
             THE NECESSARY ARRAYS THAT ARE REQUIRED TO BUILD THE SLR(1)  0005
             PARSING TABLES.  THE ARRAYS ARE NOT IN A LOGICAL           0006
             ORDER BUT IN AN ORDER THAT WOULD OPTIMIZE THE NUMBER       0007
             OF 4K DATA SEGMENTS. THESE SEGMENTS SHOULD NOT             0008
             BE CHANGED WITHOUT CHANGING THE FIRST OF THE               0009
             MAIN PROGRAM;                                              0010
                                                                        0011
    GLOBAL DATA SEGN004 BASE R11;                                       0012
     ARRAY 4096 BYTE V = ("<SYSTEMGS>_|_");                             0013
    CLOSE BASE;                                                         0014
    GLOBAL DATA SEGN005 BASE R12;                                       0015
    ARRAY 255 INTEGER LOCLENGTH =(#A, #283);                            0016
     ARRAY 2048 BYTE PRODARRAY = 0;                                     0017
    ARRAY 255 BYTE RTPTSIZE = 255(0);                                   0018
     CLOSE BASE;                                                        0019
                                                                        0020
     GLOBAL DATA SEGN006 BASE R11;                                      0021
    ARRAY 255 SHORT INTEGER PRODSTART = 0;                              0022
    ARRAY 255 SHORT INTEGER FIRSTPRODFOR;                               0023
    ARRAY 255 BYTE ONLEFT = 254(0);                                     0024
    ARRAY 255 BYTE ONRIGHT = 255(0);                                    0025
    ARRAY 255 SHORT INTEGER READTOLA = 255(0);                          0026
    ARRAY 255 SHORT INTEGER LINEARTOARRAY = 255(0);                     0027
    ARRAY 255, SHORT INTEGER TREADSTART = 255(0);                       0028
    ARRAY 255 SHORT INTEGER NTREADSTART = 255(0);                       0029
    ARRAY 255 BYTE TRDNUM = 255(0);                                     0030
    ARRAY 255 BYTE NTRDNUM = 255(0);                                    0031
    CLOSE BASE;                                                         0032
                                                                        0033
    GLOBAL DATA SEGN007 BASE R10;                                       0034
    ARRAY 1535 BYTE NTSYMLIST = 1535(0);                                0035
    ARRAY 1535 BYTE TSYMLIST = 1535(0);                                 0036
    ARRAY 255 SHORT INTEGER REDUCESUCC = 255(0);                        0037
    ARRAY 127 SHORT INTEGER SUCCSTATE;                                  0038
    ARRAY 127 SHORT INTEGER FAILSTATE;                                  0039
    CLOSE BASE;                                                         0040
                                                                        0041
    GLOBAL DATA SEBN008 BASE R12;                                       0042
```

```
ARRAY 1535 SHORT INTEGER TSTATELIST = 1535(0);                          0043
ARRAY 255 SHORT INTEGER LBSTATE = 255(0);                               0044
ARRAY 255 SHORT INTEGER RESUMESTATE;                                    0045
CLOSE BASE;                                                             0046
                                                                        0047
GLOBAL DATA SEGN009 BASE R9;                                            0048
ARRAY 1535 SHORT INTEGER NTSTATELIST = 1535(0);                         0049
ARRAY 127 BYTE LASYMNUM;                                                0050
ARRAY 127 BYTE LBSTART;                                                 0051
ARRAY 127 BYTE LBNUM;                                                   0052
ARRAY 255 SHORT INTEGER BSSTART;                                        0053
CLOSE BASE;                                                             0054
                                                                        0055
GLOBAL DATA SEGN010 BASE R8;                                            0056
ARRAY 2047 BYTE SYMBEFORE = 2047(0);                                    0057
ARRAY 511 SHORT INTEGER BASISSTACK;                                     0058
ARRAY 255 BYTE BSSIZE;                                                  0059
ARRAY 127 SHORT INTEGER CONFIGSET;                                      0060
CLOSE BASE;                                                             0061
                                                                        0062
GLOBAL DATA SEGN011 BASE R8;                                            0063
ARRAY 1023 INTEGER LOOKAHEADTABLE = 1023(0);                            0064
CLOSE BASE;                                                             0065
                                                                        0066
                                                                        0067
                                                                        0068
EQUATE READLINEAR      SYN 0;                                           0069
EQUATE READARRAY       SYN 1;                                           0070
EQUATE REDUCE          SYN 2;                                           0071
EQUATE LOOKAHEADORD    SYN 3;                                           0072
EQUATE LOOKAHEADEMPTY  SYN 4;                                           0073
EQUATE LOOKBACK        SYN 5;                                           0074
EQUATE ERRORSTATE      SYN 6;                                           0075
EQUATE EXIT            SYN 7;                                           0076
                                                                        0077
ARRAY 80 BYTE CBUF;                                                     0078
INTEGER FCBUF SYN CBUF(0);                                              0079
INTEGER SYMAFTER, CONFIG;                                               0080
INTEGER NUMREADSTATE;                                                   0081
INTEGER NUMLASTATES, NUMLBSTATES, TRDPTR, NTRDPTR;                      0082
BYTE ISLR0, ISLR1, ONESAME;                                             0083
INTEGER STATE, STATENAME, NUMCONFIGS, SN;                               0084
INTEGER SYM, BASISSETSIZE, SETSIZE, LOOKUPSTATE, NEXTSTATE;             0085
INTEGER CLENGTH, LENGTH, LOCATION, LCPT, NUMSYMS = 1, VPT = 12;         0086
INTEGER MINI, MAXI, ERRCOUNT = 0;                                       0087
BYTE READFLAG, CARDFLG, LEFTSTOP;                                       0088
ARRAY 255 BYTE CONTROL = 255(0);                                        0089
ARRAY 132 BYTE WBUF = 131(" ");                                         0090
```

```
ARRAY 132 BYTE BLANK = 132(" ");                                        0091
ARRAY 64 BYTE CBCD;                                                     0092
INTEGER NUMTERMINALS, NUMNTS, GOALSYMBOL, I, J;                         0093
INTEGER NUMPRODS = 0;                                                   0094
INTEGER REGISTER PAP SYN R10;                                           0095
ARRAY 1024 BYTE NEWV;                                                   0096
ARRAY 255 BYTE NOTDONE SYN NEWV(0);                                     0097
ARRAY 255 SHORT INTEGER READTABLE SYN NEWV(256);                        0098
BYTE ALLSAME;                                                           0099
BYTE TRUE = #FF;                                                        0100
BYTE FALSE = #00;                                                       0101
ARRAY 4 INTEGER SAVEBASEREGS;                                           0102
INTEGER NUMREDUCESTATES SYN NUMPRODS;                                   0103
INTEGER SAVE15;                                                         0104
LONG REAL CONWORK;                                                      0105
INTEGER MARK;                                                           0106
ARRAY 15 SHORT INTEGER BASISSET;                                        0107
INTEGER TREADNUM, NTREADNUM;                                            0108
INTEGER BASEREG008;                                                     0109
INTEGER BASE006;                                                        0110
INTEGER BASE005;                                                        0111
INTEGER BASE004;                                                        0112
ARRAY 2 BYTE ARROW ="->";                                               0113
ARRAY 47 BYTE DASHLINE = 47("-");                                       0114
ARRAY 25 BYTE TBUF;                                                     0115
INTEGER S;                                                              0116
INTEGER LASTATE;                                                        0117
ARRAY 90 BYTE STARLINE = 90("*");                                       0118
INTEGER MASK = #80000000;                                               0119
INTEGER NTSYM, NTSYM2;                                                  0120
BYTE ISVALID, INCL, INTER, CHANGING;                                    0121
INTEGER PRED, BASE010, BASE011;                                         0122
INTEGER VLENGTH;                                                        0123
                                                                        0124
COMMENT  DECLARE TOGGLES TO CONTROL OUTPUT;                             0125
                                                                        0126
BYTE INPUTLIST SYN CONTROL(201);                                        0127
BYTE GRAMMERLIST SYN CONTROL(199);                                      0128
BYTE CONFIGLIST SYN CONTROL(195);                                       0129
BYTE FSMLIST SYN CONTROL(198);                                          0130
BYTE LASETSLIST SYN CONTROL(211);                                       0131
BYTE DPDALIST SYN CONTROL(196);                                         0132
BYTE PUNCHDECK SYN CONTROL(215);                                        0133
BYTE MOREGRAMMERS;                                                      0134
                                                                        0135
```

```
FUNCTION SETZONE(8,#96F0);                                              0136

FUNCTION LOAD(12,#5800);                                                0137
                                                                        0138
PROCEDURE MIN(R14);                                                     0139
  BEGIN                                                                 0140
    R1 := I; R2 := J;                                                   0141
    IF R1 < R2 THEN MINI := R1                                          0142
    ELSE MINI := R2;                                                    0143
  END;                                                                  0144
                                                                        0145
PROCEDURE MAX(R14);                                                     0146
  BEGIN                                                                 0147
    R1 := I; R2 := J;                                                   0148
    IF R1 > R2 THEN MAXI := R1                                          0149
    ELSE MAXI := R2;                                                    0150
  END;                                                                  0151
                                                                        0152
PROCEDURE OUT(R14);                                                     0153
  BEGIN GOTO BAILOUT;                                                   0154
  END;                                                                  0155
                                                                        0156
PROCEDURE ERROR(R14);                                                   0157
  BEGIN                                                                 0158
    MVC(9,WBUF,"*** ERROR,");                                           0159
    R0 := @WBUF; WRITE; MVC(100,WBUF,BLANK);                            0160
    WRITE; WRITE;                                                       0161
    R1 := ERRCOUNT + 1; ERRCOUNT := R1;                                 0162
    IF R1 > 15 THEN                                                     0163
    BEGIN                                                               0164
      MVC(40,WBUF,"TOO MANY ERRORS (>15) IN THE CFSM, DPDA, ");         0165
      MVC(34,WBUF(42),"AND/OR LOOK AHEAD SETS.  EXECUTION ");           0166
      MVC(27,WBUF(76),"TERMINATED FOR THIS GRAMMAR.");                  0167
      R0 := @WBUF; WRITE; MVC(131,WBUF,BLANK);                          0168
      OUT;                                                              0169
    END;                                                                0170
  END;                                                                  0171
                                                                        0172
PROCEDURE INCNUMPRODS(R14);                                             0173
  BEGIN                                                                 0174
    R1 := NUMPRODS;                                                     0175
    IF R1 < 255 THEN                                                    0176
    BEGIN                                                               0177
      R1 := R1 + 1; NUMPRODS := R1;                                     0178
    END ELSE                                                            0179
    BEGIN                                                               0180
      MVC(20,WBUF(15),"TOO MANY PRODUCTIONS.");                         0181
```

```
      ERROR;                                                              0182
    END;                                                                  0183
  END;                                                                    0184
                                                                          0185
                                                                          0186
COMMENT  A PROCEDURE TO FIND A TOKEN IN THE ARRAY V;                      0187
                                                                          0188
PROCEDURE FIND(R14);                                                      0189
  BEGIN ARRAY 4 INTEGER SAVEREGS;                                         0190
    INTEGER SAVE14, SAVE11;                                               0191
    STM(R1,R4,SAVEREGS);                                                  0192
    SAVE14 := R14; SAVE11 := R11;                                         0193
    R11 := BASE004;                                                       0194
    MVC(63,CBCD,BLANK); R1 := VPT SHLL 2;                                 0195
    R3 := LOCLENGTH(R1); R1 := R3 SHRL 6;                                 0196
    LOCATION := R1; R1 := R3 AND  #3F;                                    0197
    LENGTH := R1; R2 := LENGTH - 1; CLENGTH := R2;                        0198
    FOR R1 := 0 STEP 1 UNTIL R2 DO                                        0199
    BEGIN                                                                 0200
      R3 := LOCATION + R1; IC(R4,V(R3)); STC(R4,CBCD(R1));                0201
    END;                                                                  0202
    LM(R1,R4,SAVEREGS);                                                   0203
    R14 := SAVE14; R11 := SAVE11;                                         0204
  END;  COMMENT END FIND;                                                 0205
                                                                          0206
                                                                          0207
                                                                          0208
GLOBAL PROCEDURE READG(R14);                                              0209
  BEGIN INTEGER SAVE14;                                                   0210
    INTEGER LP;                                                           0211
    INTEGER REGISTER CP SYN R8;                                           0212
                                                                          0213
    PROCEDURE GETCARD(R14);                                               0214
      BEGIN INTEGER SAVE14;                                               0215
        SAVE14 := R14;                                                    0216
        WHILE TRUE DO                                                     0217
        BEGIN                                                             0218
          R0 := @CBUF; READ;                                              0219
        IF ¬= THEN                                                        0220
        BEGIN                                                             0221
          RESET(MOREGRAMMERS); IF INPUTLIST THEN PAGE;                    0222
          RESET(READFLAG); GOTO ENDGETCARD;                               0223
        END;                                                              0224
        IF INPUTLIST THEN                                                 0225
        BEGIN                                                             0226
          MVC(79,WBUF(20),CBUF); R0 := @WBUF; WRITE;                      0227
          MVC(79,WBUF(20),BLANK);                                         0228
        END;                                                              0229
```

```
      R1 := R1 - R1; IC(R1,CBUF);                                          0230
      IF R1 = "@" THEN                                                     0231
      BEGIN                                                                0232
        R1 := FCBUF AND #00FFFFFF;                                         0233
        IF R1 = "EOG" THEN                                                 0234
        BEGIN                                                              0235
          IF INPUTLIST THEN PAGE;                                          0236
          RESET(READFLAG); GOTO ENDGETCARD;                                0237
        END;                                                               0238
        R1 := R1 SHRL 16 AND #FF; R2 := R2 - R2;                           0239
        IC(R2,CONTROL(R1));                                                0240
        R3 := @CONTROL(R1);                                                0241
        IF R2 > 0 THEN RESET(B3) ELSE SET(B3);                             0242
      END ELSE                                                             0243
      BEGIN                                                                0244
        CLC(79,CBUF,BLANK);                                                0245
        IF ¬= THEN SET(READFLAG);                                          0246
        SET(CARDFLG); GOTO ENDGETCARD;                                     0247
      END;                                                                 0248
      END;  COMMENT  END WHILE TRUE;                                       0249
      ENDGETCARD:                                                          0250
      CP := 0;                                                             0251
      R14 := SAVE14;                                                       0252
    END;                                                                   0253
                                                                           0254
  PROCEDURE LOOKUP(R14);                                                   0255
    BEGIN INTEGER SAVE14, SAVE11;                                          0256
    SAVE14 := R14; SAVE11 := R11;                                          0257
      R11 := BASE004;                                                      0258
      FOR R1 := 2 STEP 1 UNTIL NUMSYMS DO                                  0259
      BEGIN                                                                0260
        R2 := R1 SHLL 2; R3 := LOCLENGTH(R2) AND #3F;                      0261
        LENGTH := R3; R7 :=CLENGTH + 1;                                    0262
        IF R7 = LENGTH THEN                                                0263
        BEGIN                                                              0264
          R5 := LOCLENGTH(R2) SHRL 6;                                      0265
          R4 := @V(R5) - 1; R2 := @CBCD - 1;                               0266
          FOR R6 := 0 STEP 1 UNTIL CLENGTH DO                              0267
          BEGIN                                                            0268
            R4 := R4 + 1;  R2 := R2 + 1;  CLC(0,B4,B2);                    0269
            IF ¬= THEN GOTO NOTFOUND;                                      0270
          END;                                                             0271
          LCPT := R1; GOTO ENDLOOKUP;                                      0272
        END;                                                               0273
        NOTFOUND:                                                          0274
      END;                                                                 0275
      R1 := VPT + CLENGTH + 1; VLENGTH := R1;                              0276
      IF R1 > 4096 THEN                                                    0277
```

```
      BEGIN                                                                    0278
        MVC(31,WBUF(15),"TOO MANY INPUT CHARACTER (>4096)");                   0279
        ERROR;                                                                 0280
        MVC(37,WBUF,"EXECUTION TERMINATED FOR THIS GRAMMER.");                 0281
        R0 := @WBUF; WRITE; OUT;                                               0282
      END;                                                                     0283
      R2 := NUMSYMS + 1; NUMSYMS := R2; LCPT := R2;                            0284
      R1 := VPT + 1 SHLL 6 + CLENGTH + 1;                                      0285
      R2 := R2 SHLL 2;                                                         0286
      LOCLENGTH(R2) := R1;                                                     0287
      R1 := VPT;                                                               0288
      FOR R6 := 0 STEP 1 UNTIL CLENGTH DO                                      0289
      BEGIN                                                                    0290
        R1 := R1 + 1;  IC(R5,CBCD(R6)); STC(R5,V(R1));                         0291
      END;                                                                     0292
      VPT := R1;                                                               0293
      ENDLOOKUP : R14 := SAVE14; R11 := SAVE11;                                0294
    END;                                                                       0295
                                                                               0296
PROCEDURE DEBLANK(R14);                                                        0297
  BEGIN INTEGER SAVE14;                                                        0298
    SAVE14 := R14;                                                             0299
    WHILE R1 = 64  AND CP < 80 DO                                              0300
    BEGIN                                                                      0301
      CP := CP + 1; IC(R1,CBUF(CP)); R1 := R1 AND #FF;                         0302
    END;                                                                       0303
    R14 := SAVE14;                                                             0304
  END;                                                                         0305
                                                                               0306
PROCEDURE SCAN(R14);                                                           0307
  BEGIN INTEGER SAVE14;                                                        0308
    SAVE14 := R14;                                                             0309
    IF CP < 80 THEN                                                            0310
    BEGIN                                                                      0311
      R1 := R1 - R1; IC(R1,CBUF(CP));                                          0312
      IF R1 = 64  THEN                                                         0313
      BEGIN                                                                    0314
      DEBLANK; R1 := R1 - R1; IC(R1,CBUF(CP));                                 0315
      END;                                                                     0316
    END;                                                                       0317
    IF CP < 80 THEN                                                            0318
    BEGIN                                                                      0319
      LP := CP;                                                                0320
      IF R1  = "<" THEN SET(LEFTSTOP) ELSE RESET(LEFTSTOP);                    0321
      FOR CP := CP + 1 STEP 1 UNTIL 79 DO                                      0322
      BEGIN                                                                    0323
        R1 := R1 - R1; IC(R1,CBUF(CP));                                        0324
        IF R1 = " " AND LEFTSTOP THEN                                          0325
```

```
        BEGIN                                                          0326
          R2 := CP - LP;                                               0327
          IF R2 = 1 THEN                                               0328
          BEGIN                                                        0329
          CP := CP - 1; GOTO ENDSEARCH;                                0330
          END;                                                         0331
        END;                                                           0332
        IF R1 = " " AND ¬LEFTSTOP THEN                                 0333
        BEGIN                                                          0334
          CP := CP - 1; GOTO ENDSEARCH;                                0335
        END;                                                           0336
        IF R1 = ">" AND LEFTSTOP THEN GOTO ENDSEARCH;                  0337
      END;                                                             0338
      IF CP = 80 AND LEFTSTOP THEN                                     0339
      BEGIN                                                            0340
        MVC(17,WBUF(15),"UNMATCHED BRACKET: <");                       0341
        ERROR;                                                         0342
      END;                                                             0343
      ENDSEARCH: R2 := CP - LP; CLENGTH := R2;                         0344
      R4 := LP - 1;                                                    0345
      FOR R6 := 0 STEP 1 UNTIL CLENGTH DO                              0346
      BEGIN                                                            0347
        R4 := R4 + 1;  IC(R3,CBUF(R4)); STC(R3,CBCD(R6));              0348
      END;                                                             0349
      LOOKUP;                                                          0350
      SET(CARDFLG); GOTO ENDSCAN;                                      0351
    END;                                                               0352
    RESET(CARDFLG);                                                    0353
    ENDSCAN: CP := CP + 1; R14 := SAVE14;                              0354
  END;                                                                 0355
                                                                       0356
PROCEDURE FINDGOAL(R14);                                               0357
  BEGIN INTEGER SAVE14;                                                0358
    SAVE14 := R14; R1 := 0; GOALSYMBOL := R1;                          0359
    FOR R1 := 1 STEP 1 UNTIL NUMSYMS DO                               0360
    BEGIN                                                              0361
      R2 := R2 - R2; IC(R2,ONRIGHT(R1));                               0362
      IF R2 = 0 THEN                                                   0363
      BEGIN                                                            0364
        R3 := GOALSYMBOL;                                              0365
        IF R3 = 0 THEN GOALSYMBOL := R1                                0366
        ELSE                                                           0367
        BEGIN                                                          0368
          MVC(36,WBUF,"MORE THAN ONE GOAL SYMBOL WAS FOUND:");         0369
          R0 := @WBUF; WRITE; MVC(36,WBUF,BLANK);                      0370
          VPT := R3; FIND;                                             0371
          FOR R3 := 0 STEP 1 UNTIL CLENGTH DO                          0372
          BEGIN                                                        0373
```

```
            IC(R4,CBCD(R3)); STC(R4,WBUF(R3));                          0374
          END;                                                          0375
          MVC(4,WBUF(65),"USED,"); WRITE; MVC(69,WBUF,BLANK);           0376
          VPT := R1; FIND;                                              0377
          FOR R3 := 0 STEP 1 UNTIL CLENGTH DO                           0378
          BEGIN                                                         0379
            IC(R4,CBCD(R3)); STC(R4,WBUF(R3));                          0380
          END;                                                          0381
          MVC(7,WBUF(65),"IGNORED."); WRITE; MVC(80,WBUF,BLANK);        0382
        END;                                                            0383
      END;                                                              0384
    END;                                                                0385
    R2 := GOALSYMBOL;                                                   0386
    IF R2 = 0 THEN                                                      0387
    BEGIN                                                               0388
      MVC(33,WBUF,"NO EXPLICIT GOAL SYMBOL WAS FOUND.");                0389
      R0 := @WBUF; WRITE; MVC(33,WBUF,BLANK);                           0390
      LH(R3,PRODSTART(2)); IC(R1,PRODARRAY(R3));                        0391
      R1 := R1 AND #FF;                                                 0392
      VPT := R1; FIND;                                                  0393
      FOR R3 := 0 STEP 1 UNTIL CLENGTH DO                               0394
      BEGIN                                                             0395
        IC(R4,CBCD(R3)); STC(R4,WBUF(R3));                              0396
      END;                                                              0397
      MVC(31,WBUF(66),"WILL BE USED AS THE GOAL SYMBOL.");              0398
      WRITE; MVC(100,WBUF,BLANK); WRITE; WRITE;                         0399
    END;                                                                0400
    R1 := NUMPRODS + 1 SHLL 1; STH(PAP,PRODSTART(R1));                  0401
    R4 := 3; R5 := 0; R1 := R1 SHRL 1;                                  0402
    STC(R4 ,RTPTSIZE(R1)); STC(R5 ,PRODARRAY(PAP));                     0403
    R4 := 1;                                                            0404
    PAP := PAP + 1; STC(R4 ,PRODARRAY(PAP));                            0405
    PAP := PAP + 1; R3 := GOALSYMBOL; STC(R3,PRODARRAY(PAP));           0406
    PAP := PAP + 1; STC(R4 ,PRODARRAY(PAP));                            0407
    R14 := SAVE14;                                                      0408
  END;  COMMENT  END OF FINDGOAL;                                       0409
                                                                        0410
PROCEDURE SORTV(R14);                                                   0411
  BEGIN INTEGER SAVE14;                                                 0412
    ARRAY 255 BYTE INDEX;                                               0413
    ARRAY 255 BYTE NEWINDEX;                                            0414
    SAVE14 := R14;                                                      0415
    FOR R2 := 0 STEP 1 UNTIL NUMSYMS DO                                 0416
    BEGIN                                                               0417
      R1 := R2; STC(R1,INDEX(R2));                                      0418
    END;                                                                0419
    FOR R2 := 3 STEP 1 UNTIL NUMSYMS DO                                 0420
    BEGIN                                                               0421
```

```
  R1 := NUMSYMS;                                                    0422
  WHILE R1 >= R2 DO                                                 0423
  BEGIN                                                             0424
    R3 := R1 - 1;                                                   0425
    R4 := R4 - R4; R5 := R5 - R5;                                   0426
    IC(R4,ONLEFT(R3)); IC(R5,ONLEFT(R1));                           0427
    IF R4 = #FF AND  R5 = 0 THEN                                    0428
    BEGIN                                                           0429
      R4 := @ONLEFT(R3); R5 := @ONLEFT(R1);                         0430
      RESET(B4); SET(B5);                                           0431
      IC(R5,INDEX(R3)); IC(R6,INDEX(R1));                           0432
      STC(R6,INDEX(R3)); STC(R5,INDEX(R1));                         0433
      R3 := R3 SHLL 2; R1 := R1 SHLL 2;                             0434
      R5 := LOCLENGTH(R3); R6 := LOCLENGTH(R1);                     0435
      LOCLENGTH(R3) := R6; LOCLENGTH(R1) := R5;                     0436
      R3 := R3 SHRL 2; R1 := R1 SHRL 2;                             0437
    END;                                                            0438
    R1 := R3;                                                       0439
  END;                                                              0440
END;                                                                0441
FOR R2 := 1 STEP 1 UNTIL NUMSYMS DO                                 0442
BEGIN                                                               0443
  R1 := R1 - R1; IC(R1,INDEX(R2)); STC(R2,NEWINDEX(R1));            0444
END;                                                                0445
R2 := GOALSYMBOL; R1 := R1 - R1;                                    0446
IC(R1,NEWINDEX(R2)); GOALSYMBOL := R1;                              0447
R1 := NUMPRODS + 1 SHLL 1; R2 := PRODSTART(R1) + 3;                 0448
FOR R1 := 1 STEP 1 UNTIL R2 DO                                      0449
BEGIN                                                               0450
  R3 := R3 - R3; IC(R3,PRODARRAY(R1));                              0451
  IC(R4,NEWINDEX(R3)); STC(R4,PRODARRAY(R1));                       0452
END;                                                                0453
FOR R1 := 1 STEP 1 UNTIL NUMPRODS DO                                0454
BEGIN                                                               0455
  R3 := R1 SHLL 1;                                                  0456
  R2 := PRODSTART(R3); R3 := R3 - 2; R4 := PRODSTART(R3);           0457
  R5 := @PRODARRAY(R2); R6 := @PRODARRAY(R4);                       0458
  CLC(0,B5,B6);                                                     0459
  IF ¬= THEN                                                        0460
  BEGIN                                                             0461
    R5 := R5 - R5; IC(R5,PRODARRAY(R2));                            0462
    R5 := R5 SHLL 1;                                                0463
    FIRSTPRODFOR(R5) := R1;                                         0464
  END;                                                              0465
END;                                                                0466
R1 := 2;                                                            0467
R4 := @ONLEFT(R1);                                                  0468
CLC(0,B4,FALSE);                                                    0469
```

```
    WHILE = DO                                                          0470
    BEGIN                                                               0471
      R1 := R1 + 1; R4 := R4 + 1; CLC(0,B4,FALSE);                      0472
    END;                                                                0473
    R1 := R1 - 1;                                                       0474
    NUMTERMINALS := R1;  R2 := NUMSYMS - R1; NUMNTS := R2;              0475
    R14 := SAVE14;                                                      0476
  END;                                                                  0477
                                                                        0478
PROCEDURE PRINTG(R14);                                                  0479
  BEGIN INTEGER SAVE14;                                                 0480
    ARRAY 3 BYTE EQUAL = (":::=");                                      0481
    SAVE14 := R14;                                                      0482
    MVC(26,WBUF(53),"T H E   V O C A B U L A R Y");                     0483
    R0 := @WBUF; WRITE; MVC(26,WBUF(53),BLANK); WRITE; WRITE;           0484
    MVC(30,WBUF(10),"T E R M I N A L   S Y M B O L S");                 0485
    MVC(22,WBUF(79),"N O N T E R M I N A L S");                         0486
    WRITE; MVC(131,WBUF,BLANK); WRITE; WRITE;                           0487
    R1 := NUMTERMINALS; I := R1; R1 := NUMNTS; J := R1; MAX;            0488
    FOR R1 := 1 STEP 1 UNTIL MAXI DO                                    0489
    BEGIN                                                               0490
      CVD(R1,CONWORK); UNPK(3,7,WBUF(4),CONWORK);                       0491
      SETZONE(WBUF(7));                                                 0492
      IF R1 <= NUMTERMINALS THEN                                        0493
      BEGIN                                                             0494
        VPT := R1; FIND;                                                0495
        R5 := 9;                                                        0496
        FOR R3 := 0 STEP 1 UNTIL CLENGTH DO                             0497
        BEGIN                                                           0498
          R5 := R5 + 1;                                                 0499
          IC(R4,CBCD(R3)); STC(R4,WBUF(R5));                            0500
        END;                                                            0501
      END;                                                              0502
      IF R1 <= NUMNTS THEN                                              0503
      BEGIN                                                             0504
        R2 := R1 + NUMTERMINALS; VPT := R2; FIND;                       0505
        R4 := 79;                                                       0506
        FOR R3 := 0 STEP 1 UNTIL CLENGTH DO                             0507
        BEGIN                                                           0508
          IC(R5,CBCD(R3)); STC(R5,WBUF(R4)); R4 := R4 + 1;              0509
        END;                                                            0510
      END;                                                              0511
      WRITE; MVC(131,WBUF,BLANK);                                       0512
    END;                                                                0513
    WRITE; WRITE; MVC(17,WBUF(50),"THE GOAL SYMBOL IS");                0514
    R1 := GOALSYMBOL; VPT := R1; FIND;                                  0515
    R4 := 79;                                                           0516
    FOR R3 := 0 STEP 1 UNTIL CLENGTH DO                                 0517
```

```
BEGIN                                                                   0518
  IC(R5,CBCD(R3)); STC(R5,WBUF(R4)); R4 := R4 + 1;                      0519
END;                                                                    0520
WRITE; MVC(70,WBUF(50),BLANK); R1 := MAXI;                              0521
IF R1 > 20 THEN PAGE ELSE BEGIN WRITE; WRITE; END;                      0522
MVC(28,WBUF(25),"T H E   P R O D U C T I O N S");                       0523
WRITE; MVC(28,WBUF(25),BLANK); WRITE; WRITE;                            0524
R8 := NUMPRODS;                                                         0525
FOR R1 := 1 STEP 1 UNTIL R8 DO                                          0526
BEGIN                                                                   0527
  R1 := R1 SHLL 1;                                                      0528
  R2 := R2 - R2; R3 := PRODSTART(R1); IC(R2,PRODARRAY(R3));             0529
  R4 := R1 - 2; R5 := PRODSTART(R4); IC(R6,PRODARRAY(R5));              0530
  R6 := R6 AND #FF;                                                     0531
  IF  R2 = R6 THEN                                                      0532
  BEGIN                                                                 0533
    R4 := MARK + 2; R3 := #4F; STC(R3,WBUF(R4));                        0534
  END ELSE                                                              0535
  BEGIN                                                                 0536
    WRITE;                                                              0537
    VPT := R2; FIND; R7 := 10;                                          0538
    FOR R4 := 0 STEP 1 UNTIL CLENGTH DO                                 0539
    BEGIN                                                               0540
      IC(R5,CBCD(R4)); STC(R5,WBUF(R7)); R7 := R7 + 1;                  0541
    END;                                                                0542
    R7 := R7 + 3; MARK := R7;                                           0543
    FOR R4 := 0 STEP 1 UNTIL 2 DO                                       0544
    BEGIN                                                               0545
      IC(R5,EQUAL(R4)); STC(R5,WBUF(R7)); R7 := R7 + 1;                 0546
    END;                                                                0547
  END;                                                                  0548
  R2 := PRODSTART(R1); R3 := R2 + 1; R4 := R4 - R4;                     0549
  R1 := R1 SHRL 1;                                                      0550
  IC(R4,RTPTSIZE(R1)); R2 := R2 + R4; R5 := MARK + 5;                   0551
  FOR R3 := R3 STEP 1 UNTIL R2 DO                                       0552
  BEGIN                                                                 0553
    R4 := R4 - R4; IC(R4,PRODARRAY(R3));                                0554
    VPT := R4; FIND;                                                    0555
    FOR R4 := 0 STEP 1 UNTIL CLENGTH DO                                 0556
    BEGIN                                                               0557
      IC(R7,CBCD(R4)); STC(R7,WBUF(R5)); R5 := R5 + 1;                  0558
    END;                                                                0559
    R5 := R5 + 2;                                                       0560
  END;                                                                  0561
  CVD(R1,CONWORK); UNPK(3,7,WBUF(4),CONWORK);                           0562
  SETZONE(WBUF(7));                                                     0563
  WRITE; MVC(131,WBUF,BLANK);                                           0564
END;                                                                    0565
```

```
      PAGE;                                                                    0566
      MVC(30,WBUF,"SOME STATISTICS ON THE GRAMMAR:");                          0567
      WRITE; MVC(30,WBUF,BLANK); WRITE; WRITE;                                 0568
      MVC(28,WBUF(5),"NUMBER OF TERMINAL SYMBOLS = ");                         0569
      R1 := NUMTERMINALS; CVD(R1,CONWORK);                                     0570
      UNPK(3,7,WBUF(34),CONWORK); SETZONE(WBUF(37));                           0571
      WRITE; MVC(40,WBUF,BLANK);                                               0572
      MVC(30,WBUF(5),"NUMBER OF NONTERMINAL SYMBOLS = ");                      0573
      R1 := NUMNTS; CVD(R1,CONWORK);                                           0574
      UNPK(3,7,WBUF(37),CONWORK); SETZONE(WBUF(40));                           0575
      WRITE; MVC(50,WBUF,BLANK);                                               0576
      MVC(25,WBUF(11),"TOTAL NUMBER OF SYMBOLS = ");                           0577
      R1 := NUMTERMINALS + NUMNTS; CVD(R1,CONWORK);                            0578
      UNPK(3,7,WBUF(37),CONWORK); SETZONE(WBUF(40));                           0579
      WRITE; MVC(50,WBUF,BLANK); WRITE;                                        0580
      MVC(23,WBUF(5),"NUMBER OF PRODUCTIONS = ");                              0581
      R1 := NUMPRODS; CVD(R1,CONWORK);                                         0582
      UNPK(3,7,WBUF(29),CONWORK); SETZONE(WBUF(32));                           0583
      WRITE; MVC(40,WBUF,BLANK); PAGE;                                         0584
      R14 := SAVE14;                                                           0585
    END;  COMMENT  END OF PRINTG;                                              0586
                                                                               0587
                                                                               0588
  SAVE14 := R14; CP := 0;                                                      0589
  NUMPRODS := CP; PRODSTART(0) := CP;                                          0590
  STC(CP,PRODARRAY(0));                                                        0591
  FOR R1 := 0 STEP 1 UNTIL 254 DO                                              0592
  BEGIN                                                                        0593
    STC(CP,ONLEFT(R1)); STC(CP,ONRIGHT(R1));                                   0594
  END;                                                                         0595
  SET(ONLEFT(0)); SET(ONRIGHT(1));                                             0596
  R1 := 1; NUMSYMS := R1; R10 := R1;                                           0597
GETCARD;                                                                       0598
  WHILE READFLAG DO                                                            0599
  BEGIN                                                                        0600
    INCNUMPRODS; CLC(0,CBUF,BLANK);                                            0601
    IF = THEN                                                                  0602
    BEGIN                                                                      0603
      R9 := R9 - R9; R2 := NUMPRODS - 1 SHLL 1;                                0604
      R3 := PRODSTART(R2); IC(R9,PRODARRAY(R3));                               0605
    END ELSE                                                                   0606
    BEGIN                                                                      0607
      SCAN; R9 := LCPT AND #FF;                                                0608
    END;                                                                       0609
    IC(R2,TRUE); STC(R2,ONLEFT(R9));                                           0610
    R3 := NUMPRODS SHLL 1; STH(PAP,PRODSTART(R3));                             0611
    R3 := _1; R4 := NUMPRODS; STC(R3,RTPTSIZE(R4));                            0612
    WHILE CARDFLG DO                                                           0613
```

```
        BEGIN                                                                   0614
          STC(R9,PRODARRAY(PAP)); PAP := PAP + 1;                               0615
          R4 := NUMPRODS;                                                       0616
          IC(R3,RTPTSIZE(R4)); R3 := R3 + 1; STC(R3,RTPTSIZE(R4));              0617
          SCAN; R9 := LCPT; R2 := #FF; STC(R2,ONRIGHT(R9));                     0618
        END;                                                                    0619
        R3 := ERRCOUNT;                                                         0620
        IF R3 > 10 THEN                                                         0621
        BEGIN                                                                   0622
          MVC(15,WBUF,"TOO MANY ERRORS.");                                      0623
          MVC(36,WBUF(18),"EXECUTION TERMINATED FOR THIS GRAMMAR");             0624
          R0 := @WBUF; WRITE; MVC(52,WBUF,BLANK);                               0625
          WHILE READFLAG DO GETCARD;                                            0626
        END;                                                                    0627
        GETCARD;                                                                0628
      END;                                                                      0629
      FINDGOAL;                                                                 0630
      SORTV;                                                                    0631
      R1 := ERRCOUNT;                                                           0632
      IF GRAMMERLIST OR R1 ¬= 0 THEN PRINTG;                                    0633
      R14 := SAVE14;                                                            0634
    END;  COMMENT  END OF READG;                                                0635
                                                                                0636
                                                                                0637
                                                                                0638
GLOBAL PROCEDURE COMPUTECFSM(R14);                                              0639
  BEGIN INTEGER SAVE14; ARRAY 7 INTEGER SAVEREGS;                               0640
                                                                                0641
    PROCEDURE SYMAFTERDOT(R14);                                                 0642
      BEGIN INTEGER SAVE14; ARRAY 4 INTEGER SAVEREGS;                           0643
        INTEGER SAVE12;                                                         0644
        SAVE12 := R12; R12 := BASE005;                                          0645
        SAVE14 := R14; STM(R1,R4,SAVEREGS);                                     0646
        R1 := CONFIG SHRL 8 AND #FF;                                            0647
        R3 := R3 - R3;                                                          0648
        R2 := CONFIG AND #FF; IC(R3,RTPTSIZE(R2));                              0649
        IF R1 < R3 THEN                                                         0650
        BEGIN                                                                   0651
          R2 := R2 SHLL 1; R3 := PRODSTART(R2); R4 := R4 - R4;                  0652
          R3 := R3 + R1 + 1; IC(R4,PRODARRAY(R3)); SYMAFTER := R4;              0653
        END ELSE                                                                0654
        BEGIN                                                                   0655
          R2 := 0; SYMAFTER := R2;                                              0656
        END;                                                                    0657
        LM(R1,R4,SAVEREGS); R14 := SAVE14; R12 := SAVE12;                       0658
      END;  COMMENT  END OF SYMAFTER DOT;                                       0659
                                                                                0660
    PROCEDURE INCNUMCONFIGS(R14);                                               0661
```

```
  BEGIN INTEGER SAVE1, SAVE14;                                          0662
    SAVE1 := R1; SAVE14 := R14; R1 := NUMCONFIGS + 1;                   0663
    IF R1 < 128 THEN NUMCONFIGS := R1                                   0664
    ELSE BEGIN                                                          0665
      MVC(30,WBUF(15),"THE CONFIGURATION SET FOR STATE");               0666
      R1 := STATE; CVD(R1,CONWORK);  UNPK(3,7,WBUF(47),CONWORK);        0667
      SETZONE(WBUF(50)); MVC(12,WBUF(53),"IS TOO LARGE.");              0668
      ERROR;  SET(CONFIGLIST);                                          0669
    END;                                                                0670
    R1 := SAVE1; R14 := SAVE14;                                         0671
  END;  COMMENT END OF INCNUMCONFIGS;                                   0672
                                                                        0673
PROCEDURE INCTRDPTR(R14);                                               0674
  BEGIN INTEGER SAVE1, SAVE14;                                          0675
    SAVE1 := R1; SAVE14 := R14; R1 := TRDPTR + 1;                       0676
    IF R1 < 1536 THEN TRDPTR := R1                                      0677
    ELSE BEGIN                                                          0678
      MVC(28,WBUF(15),"TOO MANY TERMINAL TRANSITIONS");                 0679
      ERROR;                                                            0680
    END;                                                                0681
    R1 := SAVE1; R14 := SAVE14;                                         0682
  END;  COMMENT  NEND OF INCTRDPTR;                                     0683
                                                                        0684
PROCEDURE INCNTRDPTR(R14);                                              0685
  BEGIN INTEGER SAVE1, SAVE14;                                          0686
    SAVE1 := R1; SAVE14 := R14; R1 := NTRDPTR + 1;                      0687
    IF R1 < 1536 THEN NTRDPTR := R1                                     0688
    ELSE BEGIN                                                          0689
      MVC(31,WBUF(15),"TOO MANY NONTERMINAL TRANSITIONS");              0690
      ERROR;                                                            0691
    END;                                                                0692
    R14 := SAVE14; R1 := SAVE1;                                         0693
  END;  COMMENT END OF INCNTRDPTR;                                      0694
                                                                        0695
PROCEDURE INCNUMLASTATES(R14);                                          0696
  BEGIN  INTEGER SAVE1, SAVE14;                                         0697
    SAVE1 := R1; SAVE14 := R14; R1 := NUMLASTATES + 1;                  0698
    IF R1 < 128 THEN NUMLASTATES := R1                                  0699
    ELSE BEGIN                                                          0700
      MVC(25,WBUF(15),"TOO MANY LOOK AHEAD STATES");                    0701
      ERROR;                                                            0702
  END;                                                                  0703
    R1 := SAVE1; R14 := SAVE14;                                         0704
  END;  COMMENT  END OF INCNUMLASTATES;                                 0705
                                                                        0706
PROCEDURE PRINTCS(R14);                                                 0707
  BEGIN INTEGER SAVE14,DOT;                                             0708
    SAVE14 := R14;                                                      0709
```

```
MVC(30,WBUF,"THE CONFIGURATION SET FOR STATE");                    0710
R1 := STATE; CVD(R1,CONWORK);                                      0711
UNPK(3,7,WBUF(35),CONWORK); SETZONE(WBUF(38));                     0712
MVC(2,WBUF(40),"IS:"); R0 := @WBUF; WRITE;                         0713
MVC(45,WBUF,BLANK);                                                0714
FOR R1 := 1 STEP 1 UNTIL NUMCONFIGS DO                             0715
BEGIN                                                              0716
  R2 := R1 SHLL 1; R3 := CONFIGSET(R2) AND #FF;                    0717
  R3 := R3 SHLL 1; R2 := PRODSTART(R3);                            0718
  R3 := R3 - R3; IC(R3,PRODARRAY(R2));                             0719
  VPT := R3; FIND; R4 := 10;                                       0720
  FOR R6 := 0 STEP 1 UNTIL CLENGTH DO                              0721
  BEGIN                                                            0722
    IC(R5,CBCD(R6)); STC(R5,WBUF(R4)); R4 := R4 + 1;               0723
  END;                                                             0724
  R2 := R4 + 3; R4 := R4 + 2;                                      0725
  FOR R6 := 0 STEP 1 UNTIL 1 DO                                    0726
  BEGIN                                                            0727
    IC(R5,ARROW(R6)); STC(R5,WBUF(R4)); R4 := R4 + 1;              0728
  END;                                                             0729
  R1 := R1 SHLL 1; R3 := CONFIGSET(R1) SHRL 8;                     0730
  DOT := R3; R1 := R1 SHRL 1;                                      0731
  IF R3 = 0 THEN                                                   0732
  BEGIN                                                            0733
    R4 := R4 + 1; IC(R5,"."); STC(R5,WBUF(R4));                    0734
  END;                                                             0735
  R1 := R1 SHLL 1;                                                 0736
  R5 := R5 - R5; R3 := CONFIGSET(R1) AND #FF;                      0737
  R1 := R1 SHRL 1; IC(R5,RTPTSIZE(R3));                            0738
  R3 := R3 SHLL 1;                                                 0739
  FOR R2 := 1 STEP 1 UNTIL R5 DO                                   0740
  BEGIN                                                            0741
    R4 := R4 + 2;                                                  0742
    R6 := PRODSTART(R3) + R2; R7 := R7 - R7;                       0743
    IC(R7,PRODARRAY(R6)); VPT := R7; FIND;                         0744
    FOR R7 := 0 STEP 1 UNTIL CLENGTH DO                            0745
    BEGIN                                                          0746
      IC(R6,CBCD(R7)); STC(R6,WBUF(R4)); R4 := R4 + 1;             0747
    END;                                                           0748
    IF R2 = DOT THEN                                               0749
    BEGIN                                                          0750
      R4 := R4 + 1; IC(R6,"."); STC(R6,WBUF(R4));                  0751
    END;                                                           0752
  END;                                                             0753
  CVD(R1,CONWORK); UNPK(3,7,WBUF(4),CONWORK);                      0754
  SETZONE(WBUF(7)); WRITE; MVC(131,WBUF,BLANK);                    0755
END;                                                               0756
WRITE; WRITE; R14 := SAVE14;                                       0757
```

```
  END;  COMMENT  END OF PRINTCS;                                                0758
                                                                               0759
PROCEDURE ADDSUCCTOCS(R14);                                                    0760
  BEGIN INTEGER SAVE14; ARRAY 5 INTEGER SAVEREGS;                              0761
    SAVE14 := R14; STM(R1,R5,SAVEREGS);                                        0762
    R2 := SYM SHLL 1; R3 := FIRSTPRODFOR(R2);                                  0763
    R1 := R3 SHLL 1; R4 := PRODSTART(R1); R5 := R5 - R5;                       0764
    IC(R5,PRODARRAY(R4)); R2 := 0;                                             0765
    WHILE R5 = SYM DO                                                          0766
    BEGIN                                                                      0767
      INCNUMCONFIGS; R1 := NUMCONFIGS SHLL 1; R4 := R3 + R2;                   0768
      STH(R4,CONFIGSET(R1)); R2 := R2 + 1; R1 := R3 + R2 SHLL 1;               0769
      R4 := PRODSTART(R1); R5 := R5 - R5; IC(R5,PRODARRAY(R4));                0770
    END;                                                                       0771
    LM(R1,R5,SAVEREGS); R14 := SAVE14;                                         0772
  END;  COMMENT  END OF ADDSUCCTOCS;                                           0773
                                                                               0774
PROCEDURE SORTCS(R14);                                                         0775
  BEGIN INTEGER TEMP,SAVE14, HOLD;                                             0776
    SAVE14 := R14; R2 := NUMCONFIGS - 1; TEMP := R2;                           0777
    FOR R1 := 0 STEP 1 UNTIL TEMP DO                                           0778
    BEGIN                                                                      0779
      IF R1 = 0 THEN SYM := R1                                                 0780
    ELSE BEGIN                                                                 0781
        R1 := R1 SHLL 1; R2 := CONFIGSET(R1); CONFIG := R2;                    0782
        SYMAFTERDOT; R2 := SYMAFTER;  SYM := R2;                               0783
        R1 := R1 SHRL 1;                                                       0784
      END;                                                                     0785
      R3 := R1 + 1 SHLL 1; R4 := CONFIGSET(R3);                                0786
      CONFIG := R4; SYMAFTERDOT; R2 := SYM;                                    0787
      IF R2 ¬= SYMAFTER THEN                                                   0788
      FOR R3 := R1 + 2 STEP 1 UNTIL NUMCONFIGS DO                              0789
      BEGIN                                                                    0790
        R4 := R3 SHLL 1; R2 := CONFIGSET(R4); CONFIG := R2;                    0791
        SYMAFTERDOT; R4 := SYM;                                                0792
        IF R4 = SYMAFTER THEN                                                  0793
        BEGIN                                                                  0794
          HOLD := R2;                                                          0795
          R4 := R3; R5 := R1 + 1;                                              0796
          WHILE R4 > R5 DO                                                     0797
          BEGIN                                                                0798
            R7 := R4 - 1 SHLL 1; R6 := CONFIGSET(R7);                          0799
            R7 := R4 SHLL 1; STH(R6,CONFIGSET(R7)); R4 := R4 -1;               0800
          END;                                                                 0801
          R5 := HOLD;                                                          0802
          R7 := R4 SHLL 1; STH(R5,CONFIGSET(R7)); R1 := R4;                    0803
        END;                                                                   0804
      END;                                                                     0805
```

```
    END;                                                                    0806
    R14 := SAVE14;                                                          0807
  END;  COMMENT  END OF SORTCS;                                             0808
                                                                            0809
PROCEDURE COMPUTECONFIGSET(R14);                                            0810
  BEGIN INTEGER SAVE14; ARRAY 7 INTEGER SAVEREGS;                           0811
    INTEGER SAVE12;                                                         0812
    SAVE12 := R12; R12 := BASE005;                                          0813
    SAVE14 := R14; STM(R1,R7,SAVEREGS);                                     0814
    R2 := R2 - R2; R1 := STATE; IC(R2,BSSIZE(R1));                          0815
    R2 := R2 - 1; R1 := R1 SHLL 1; R4 := BSSTART(R1);                       0816
    FOR R3 := 0 STEP 1 UNTIL R2 DO                                          0817
    BEGIN                                                                   0818
      R5 := R4 + R3 SHLL 1; R6 := BASISSTACK(R5);                           0819
      R5 := R3 + 1 SHLL 1; STH(R6,CONFIGSET(R5));                           0820
    END;                                                                    0821
    NUMCONFIGS := R3; R1 := NUMTERMINALS + 1;                               0822
    R2 := NUMTERMINALS + NUMNTS; R3 := @NOTDONE(R1);                        0823
    FOR R1 := R1 STEP 1 UNTIL R2 DO                                         0824
    BEGIN                                                                   0825
      SET(B3); R3 := R3 + 1;                                                0826
    END;                                                                    0827
    R1 := 1;                                                                0828
    WHILE R1 <= NUMCONFIGS DO                                               0829
    BEGIN                                                                   0830
      R2 := R1 SHLL 1; R3 := CONFIGSET(R2); CONFIG := R3;                   0831
      SYMAFTERDOT; R2 := SYMAFTER; R3 := R3 - R3;                           0832
      R4 := R4 - R4; IC(R3,ONLEFT(R2)); IC(R4,NOTDONE(R2));                 0833
      IF R3 ¬= 0 AND R4 ¬= 0 AND R2 ¬= 0 THEN                               0834
      BEGIN                                                                 0835
        SYM := R2; ADDSUCCTOCS; R5 := @NOTDONE(R2); RESET(B5);              0836
      END;                                                                  0837
      R1 := R1 + 1;                                                         0838
    END;                                                                    0839
    R1 := NUMCONFIGS;                                                       0840
    IF R1 > 1 THEN SORTCS;                                                  0841
    IF CONFIGLIST THEN PRINTCS;                                             0842
    R14 := SAVE14; LM(R1,R7,SAVEREGS); R12 := SAVE12;                       0843
  END;  COMMENT  END OF COMPUTECONFIGSET;                                   0844
                                                                            0845
PROCEDURE INCBASISSETSIZE(R14);                                             0846
  BEGIN INTEGER SAVE1,SAVE14;                                               0847
    SAVE1 := R1; SAVE14 := R14; R1 := BASISSETSIZE + 1;                     0848
    IF R1 < 16 THEN BASISSETSIZE := R1                                      0849
    ELSE BEGIN                                                              0850
      MVC(22,WBUF(15),"THE BASIS SET FOR STATE");                           0851
      R1 := STATE;CVD(R1,CONWORK); UNPK(3,7,WBUF(39),CONWORK);              0852
      SETZONE(WBUF(42)); MVC(9,WBUF(45),"IS TOO BIG");                      0853
```

```
      ERROR;                                                                    0854
    END;                                                                        0855
    R1 := SAVE1; R14 := SAVE14;                                                 0856
  END;                                                                          0857
                                                                                0858
PROCEDURE LOOKUPREAD(R14);                                                      0859
  BEGIN INTEGER SAVE14; ARRAY 7 INTEGER SAVEREGS;                               0860
    SAVE14 := R14; STM(R1,R7,SAVEREGS);                                         0861
    FOR R1 := 1 STEP 1 UNTIL NUMREADSTATES DO                                   0862
    BEGIN                                                                       0863
      R2 := R2 - R2; IC(R2,BSSIZE(R1));                                         0864
      IF R2 = SETSIZE THEN                                                      0865
      BEGIN                                                                     0866
        SET(ALLSAME);                                                           0867
        R2 := R1 SHLL 1; R3 := BSSTART(R2);                                     0868
        R4 := R3 + SETSIZE - 1;                                                 0869
        FOR R3 := R3 STEP 1 UNTIL R4 DO                                         0870
        BEGIN                                                                   0871
          RESET(ONESAME);                                                       0872
          R2 := R3 SHLL 1; R6 := BASISSTACK(R2);                                0873
          FOR R2 := 1 STEP 1 UNTIL SETSIZE DO                                   0874
          BEGIN                                                                 0875
            R7 := R2 SHLL 1;                                                    0876
            IF R6 = BASISSET(R7) THEN SET(ONESAME);                             0877
          END;                                                                  0878
          NC(0,ALLSAME,ONESAME);                                                0879
        END;                                                                    0880
        IF ALLSAME THEN                                                         0881
        BEGIN                                                                   0882
          LASTATE := R1; GOTO ENDLOOKUPREAD;                                    0883
        END;                                                                    0884
      END;                                                                      0885
    END;                                                                        0886
    LASTATE := R1; R1 := NUMREADSTATES + 1;                                     0887
    IF R1 < 256 THEN NUMREADSTATES := R1                                        0888
    ELSE BEGIN                                                                  0889
      MVC(19,WBUF(15),"TOO MANY READ STATES");                                  0890
      ERROR;                                                                    0891
    END;                                                                        0892
    R1 := LASTATE - 1; R3 := R1 SHLL 1; R2 := R2 - R2;                          0893
    IC(R2,BSSIZE(R1)); R2 := R2 + BSSTART(R3); R3 := R3 + 2;                    0894
    BSSTART(R3) := R2; R3 := R3 SHRL 1; R1 := SETSIZE;                          0895
    STC(R1,BSSIZE(R3)); R1 := R1 + R2;                                          0896
    IF R1 > 511 THEN                                                            0897
    BEGIN                                                                       0898
      MVC(32,WBUF(15),"CFSM(THE SET OF BASIS SET)IS TOO LARGE");                0899
      ERROR; R1 := 512 - SETSIZE; R2 := LASTATE SHLL 1;                         0900
      STH(R1,BSSTART(R2));                                                      0901
```

```
    END;                                                                0902
    R2 := LASTATE SHLL 1; R1 := BSSTART(R2) - 1 SHLL 1;                 0903
    R4 := SETSIZE SHLL 1;                                               0904
    FOR R3 := 2 STEP 2 UNTIL R4 DO                                      0905
    BEGIN                                                               0906
      R6 := R1 + R3;                                                    0907
      R5 := BASISSET(R3); STH(R5,BASISSTACK(R6));                       0908
    END;                                                                0909
    ENDLOOKUPREAD:                                                      0910
    LM(R1,R7,SAVEREGS); R14 := SAVE14;                                  0911
  END;  COMMENT  END LOOKUPREAD;                                        0912
                                                                        0913
PROCEDURE ADDREADXITION(R14);                                           0914
  BEGIN  ARRAY 4 INTEGER SAVEREGS;                                      0915
    STM(R1,R4,SAVEREGS); R1 := SYM; R2 := NEXTSTATE;                    0916
    R3 := @ONLEFT(R1);                                                  0917
    CLC(0,TRUE,B3);                                                     0918
    IF = THEN                                                           0919
    BEGIN                                                               0920
      R4 := NTREADNUM + 1; NTREADNUM := R4;                             0921
    END ELSE                                                            0922
    BEGIN                                                               0923
      R4 := TREADNUM + 1; TREADNUM := R4;                               0924
    END;                                                                0925
    R1 := R1 SHLL 1; STH(R2,READTABLE(R1));                             0926
    LM(R1,R4,SAVEREGS);                                                 0927
  END;  COMMENT  END OF ADDREADXITION;                                  0928
                                                                        0929
PROCEDURE ENCODEREAD(R14);                                              0930
  BEGIN INTEGER SAVE14; ARRAY 7 INTEGER SAVEREGS;                       0931
    INTEGER SAVE12;                                                     0932
    SAVE14 := R14; STM(R1,R7,SAVEREGS);                                 0933
    SAVE12 := R12; R12 := BASEREG008;                                   0934
    R1 := TREADNUM; R3 := R1 * 3;                                       0935
    IF R3 < NUMTERMINALS AND R1 < 16 THEN                               0936
    BEGIN                                                               0937
      R3 := TRDPTR; R2 := STATE SHLL 1;                                 0938
      STH(R3,TREADSTART(R2)); R2 := R2 SHRL 1;                          0939
      STC(R1,TRDNUM(R2));                                               0940
      FOR R4 := 1 STEP 1 UNTIL NUMTERMINALS DO                          0941
      BEGIN                                                             0942
        R2 := R4 SHLL 1; R5 := READTABLE(R2);                           0943
        IF R5 ¬= #6FF THEN                                              0944
        BEGIN                                                           0945
          R3 := TRDPTR;                                                 0946
          STC(R4,TSYMLIST(R3)); R3 := R3 SHLL 1;                        0947
          STH(R5,TSTATELIST(R3)); INCTRDPTR;                            0948
        END;                                                            0949
```

```
      END;                                                              0950
      R4 := ERRORSTATE SHLL 8 OR 255;                                   0951
      R3 := TRDPTR SHLL 1; STH(R4,TSTATELIST(R3));                      0952
      INCTRDPTR;                                                        0953
    END ELSE                                                            0954
    BEGIN                                                               0955
      R2 := STATE; R1 := READARRAY SHLL 8 OR R2;                        0956
      R3 := R2 SHLL 1; STH(R1,LINEARTOARRAY(R3)); R4 := TRDPTR;         0957
      STH(R4,TREADSTART(R3)); R4 := NUMTERMINALS + 1;                   0958
      STC(R4,TRDNUM(R2));                                               0959
      FOR R4 := 0 STEP 1 UNTIL NUMTERMINALS DO                          0960
      BEGIN                                                             0961
        R2 := TRDPTR + R4; STC(R4,TSYMLIST(R2));                        0962
        R3 := R4 SHLL 1; R5 := READTABLE(R3);  R2 := R2 SHLL 1;         0963
        STH(R5,TSTATELIST(R2));                                         0964
      END;                                                              0965
      R2 := TRDPTR + NUMTERMINALS; TRDPTR := R2;                        0966
      INCTRDPTR;                                                        0967
    END;                                                                0968
    R2 := NTRDPTR; R1 := STATE SHLL 1;                                  0969
    STH(R2,NTREADSTART(R1)); R1 := R1 SHRL 1;                           0970
    R5 := NTREADNUM; STC(R5,NTRDNUM(R1));                               0971
    R5 := NUMTERMINALS + NUMNTS;                                        0972
    FOR R1 := NUMTERMINALS + 1 STEP 1 UNTIL R5 DO                       0973
    BEGIN                                                               0974
      R2 := NTRDPTR; R3 := R1 SHLL 1; R4 := READTABLE(R3);              0975
      IF R4 ¬= #6FF THEN                                                0976
      BEGIN                                                             0977
        STC(R1,NTSYMLIST(R2)); R2 := R2 SHLL 1;                         0978
        STH(R4,NTSTATELIST(R2)); INCNTRDPTR;                            0979
      END;                                                              0980
    END;                                                                0981
    LM(R1,R7,SAVEREGS); R14 := SAVE14; R12 := SAVE12;                   0982
  END;  COMMENT  END OF ENCODEREAD;                                     0983
                                                                        0984
PROCEDURE ADDSUCCESSORSTOBS(R14);                                       0985
  BEGIN INTEGER SAVE14; ARRAY 7 INTEGER SAVEREGS;                       0986
    INTEGER SAVE12;                                                     0987
    SAVE14 := R14; STM(R1,R7,SAVEREGS);                                 0988
    SAVE12 := R12; R12 := BASE005;                                      0989
    R1 := 0; BASISSETSIZE := R1;                                        0990
    FOR R2 := 0 STEP 2 UNTIL 28 DO STH(R1,BASISSET(R2));                0991
    R1 := 1;                                                            0992
    R2 := R1 SHLL 1; R3 := CONFIGSET(R2); CONFIG := R3;                 0993
    SYMAFTERDOT; R2 := SYMAFTER;                                        0994
    WHILE R1 <= NUMCONFIGS AND R2 = 0 DO                                0995
    BEGIN                                                               0996
      R3 := R3 AND #FF; R4 := R4 - R4; IC(R4,RTPTSIZE(R3));             0997
```

```
  IF R4 ¬= 0 THEN R5 := LOOKAHEADORD                                    0998
  ELSE R5 := LOOKAHEADEMPTY;                                            0999
  R5 := R5 SHLL 8 OR NUMLASTATES;                                       1000
  IF R1 = 1 THEN                                                        1001
  BEGIN                                                                 1002
    R2 := STATE SHLL 1;                                                 1003
    RESET(ISLR0); STH(R5,READTOLA(R2));                                 1004
  END;                                                                  1005
  R2 := NUMLASTATES; R6 := R3 SHLL 1;                                   1006
  R7 := PRODSTART(R6); IC(R6,PRODARRAY(R7));                            1007
  STC(R6,LASYMNUM(R2));                                                 1008
  R6 := REDUCE SHLL 8 OR R3; R2 := R2 SHLL 1;                           1009
  STH(R6,SUCCSTATE(R2));                                                1010
  IF R1 > 1 THEN                                                        1011
  BEGIN                                                                 1012
    R2 := R2 - 2;  STH(R5,FAILSTATE(R2));                               1013
  END;                                                                  1014
  R2 := STATE; IC(R4,SYMBEFORE(R2)); STC(R4,SYMBEFORE(R5));             1015
  INCNUMLASTATES; R1 := R1 + 1;                                         1016
  R2 := R1 SHLL 1; R3 := CONFIGSET(R2); CONFIG := R3;                   1017
  SYMAFTERDOT; R2 := SYMAFTER;                                          1018
END;                                                                    1019
IF R1 > 1 THEN                                                          1020
BEGIN                                                                   1021
IF R1 > NUMCONFIGS THEN                                                 1022
BEGIN                                                                   1023
  R2 := NUMLASTATES - NUMCONFIGS;                                       1024
  R3 := ERRORSTATE SHLL 8 OR R2;                                        1025
  R2 := NUMLASTATES - 1 SHLL 1; STH(R3,FAILSTATE(R2));                  1026
END ELSE                                                                1027
BEGIN                                                                   1028
  R3 := STATE; R2 := NUMLASTATES - 1 SHLL 1;                            1029
  STH(R3,FAILSTATE(R2));                                                1030
END;                                                                    1031
END;                                                                    1032
R2 := NUMTERMINALS + NUMNTS SHLL 1; R3 := #6FF;                         1033
FOR R4 := 0 STEP 2 UNTIL R2 DO STH(R3,READTABLE(R4));                   1034
R4 := 0;TREADNUM := R4; NTREADNUM := R4;                                1035
WHILE R1 <= NUMCONFIGS DO                                               1036
BEGIN                                                                   1037
  R4 := 0; BASISSETSIZE := R4;                                          1038
  R2 := R1 SHLL 1; R3 := CONFIGSET(R2); CONFIG := R3;                   1039
  SYMAFTERDOT; R5 := SYMAFTER; SYM := R5;                               1040
  WHILE R5 = SYMAFTER AND R1 <= NUMCONFIGS DO                           1041
  BEGIN                                                                 1042
    INCBASISSETSIZE; R2 := BASISSETSIZE SHLL 1;                         1043
    R4 := R3 + 256; STH(R4,BASISSET(R2)); R1 := R1 + 1;                 1044
    R2 := R1 SHLL 1; R3 := CONFIGSET(R2); CONFIG := R3;                 1045
```

```
        SYMAFTERDOT;                                                    1046
      END;                                                              1047
      R2 := BASISSETSIZE; R3 := BASISSET(2); CONFIG := R3;              1048
      SYMAFTERDOT; R3 := SYMAFTER;                                      1049
      IF R2 = 1 AND R3 = 0 THEN                                         1050
      BEGIN                                                             1051
        R3 := BASISSET(2) AND #FF;                                      1052
        R5 := REDUCE SHLL 8 OR R3;                                      1053
      END ELSE                                                          1054
      BEGIN                                                             1055
        SETSIZE := R2; LOOKUPREAD; R5 := LASTATE;                       1056
      END;                                                              1057
      NEXTSTATE := R5; ADDREADXITION;                                   1058
      R2 := SYM; STC(R2,SYMBEFORE(R5));                                 1059
      IF R1 > NUMCONFIGS THEN ENCODEREAD;                               1060
    END;                                                                1061
    R14 := SAVE14; LM(R1,R7,SAVEREGS); R12 := SAVE12;                   1062
  END;  COMMENT  END OF ADDSUCCESSORSTOBS;                              1063
                                                                        1064
                                                                        1065
PROCEDURE CONVERT(R14);                                                 1066
  BEGIN ARRAY 3 INTEGER SAVEREGS; INTEGER SAVE14;                       1067
    STM(R1,R3,SAVEREGS); SAVE14 := R14;                                 1068
    R1 := STATENAME SHLL 1; R2 := READTOLA(R1);                         1069
    IF R2 > 767 THEN SN := R2                                           1070
    ELSE BEGIN                                                          1071
      R2 := R2 SHLL 1; R3 := LINEARTOARRAY(R2); SN := R3;               1072
    END;                                                                1073
    LM(R1,R3,SAVEREGS); R14 := SAVE14;                                  1074
  END;  COMMENT  END OF CONVERT;                                        1075
                                                                        1076
STM(R1,R7,SAVEREGS); SAVE14 := R14;                                     1077
INCNUMPRODS;                                                            1078
R1 := NUMPRODS SHLL 1; R2 := PRODSTART(R1);                             1079
R1 := 0; NUMREADSTATE := R1; TRDPTR := R1; NTRDPTR := R1;               1080
STC(R1,PRODARRAY(R2));                                                  1081
R12 := BASEREG008;                                                      1082
NUMLASTATES := R1; R2 := 1; R3 := NUMREADSTATES;                        1083
STC(R2,BSSIZE(R3)); R3 := R3 SHLL 1; STH(R1,BSSTART(R3));               1084
R2 := NUMPRODS;          STH(R2,BASISSTACK(0));                         1085
WHILE R1 <= NUMREADSTATES DO                                            1086
BEGIN                                                                   1087
  STATE := R1; R2 := R1 SHLL 1; STH(R1,READTOLA(R2));                   1088
  STH(R1,LINEARTOARRAY(R2)); COMPUTECONFIGSET;                          1089
  ADDSUCCESSORSTOBS; R1 := R1 + 1;                                      1090
END;                                                                    1091
R2 := NUMPRODS SHLL 1; R3 := EXIT SHLL 8;                               1092
STH(R3,REDUCESUCC(R2));                                                 1093
```

```
    FOR R1 := 0 STEP 1 UNTIL NUMREADSTATES DO                                    1094
    BEGIN                                                                        1095
      R2 := R1 SHLL 1; R3 := TREADSTART(R2);                                     1096
      R4 := R4 - R4; IC(R4,TRDNUM(R1));                                          1097
      R4 := R4 + R3 - 1 SHLL 1;                                                  1098
      FOR R3 := R3 SHLL 1 STEP 2 UNTIL R4 DO                                     1099
      BEGIN                                                                      1100
        R2 := TSTATELIST(R3);                                                    1101
        IF R2 <= 255 THEN                                                        1102
        BEGIN                                                                    1103
          STATENAME := R2; CONVERT; R2 := SN;                                    1104
          STH(R2,TSTATELIST(R3));                                                1105
        END;                                                                     1106
      END;                                                                       1107
      R2 := R1 SHLL 1; R3 := NTREADSTART(R2);                                    1108
      R4 := R4 - R4; IC(R4,NTRDNUM(R1));                                         1109
      R4 := R4 + R3 - 1 SHLL 1;                                                  1110
      FOR R3 := R3 SHLL 1 STEP 2 UNTIL R4 DO                                     1111
      BEGIN                                                                      1112
        R2 := NTSTATELIST(R3);                                                   1113
        IF R2 <= 255 THEN                                                        1114
        BEGIN                                                                    1115
          STATENAME := R2; CONVERT; R2 := SN;                                    1116
          STH(R2,NTSTATELIST(R3));                                               1117
        END;                                                                     1118
      END;                                                                       1119
    END;                                                                         1120
    R7 := NUMLASTATES - 1 SHLL 1;                                                1121
    FOR R1 := 0 STEP 2 UNTIL R7 DO                                               1122
    BEGIN                                                                        1123
      R2 := FAILSTATE(R1);                                                       1124
      IF R2 < 256 THEN                                                           1125
      BEGIN                                                                      1126
        R2 := R2 SHLL 1;                                                         1127
        R3 := LINEARTOARRAY(R2); STH(R3,FAILSTATE(R1));                          1128
      END;                                                                       1129
    END;                                                                         1130
    LM(R1,R7,SAVEREGS); R14 := SAVE14; R12 := BASE005;                           1131
  END;  COMMENT  END OF COMPUTECFSM;                                             1132
                                                                                 1133
                                                                                 1134
                                                                                 1135
PROCEDURE CONVSTATE(R14);                                                        1136
  BEGIN INTEGER SAVE14;                                                          1137
    SAVE14 := R14; MVC(24,TBUF,BLANK);                                           1138
    R14 := S SHRL 8 + 1;                                                         1139
    CASE R14 OF                                                                  1140
    BEGIN                                                                        1141
```

```
        MVC(10,TBUF,"READ LINEAR");                                          1142
        MVC(9,TBUF,"READ ARRAY");                                            1143
        MVC(5,TBUF,"REDUCE");                                                1144
        MVC(13,TBUF,"LOOK AHEAD ORD");                                       1145
        MVC(15,TBUF,"LOOK AHEAD EMPTY");                                     1146
        MVC(8,TBUF,"LOOK BACK");                                             1147
        MVC(10,TBUF,"ERROR (FROM");                                          1148
        MVC(3,TBUF,"EXIT");                                                  1149
      END;                                                                   1150
      R14 := S AND #FF;                                                      1151
      CVD(R14,CONWORK); UNPK(3,7,TBUF(18),CONWORK);                          1152
      SETZONE(TBUF(21));                                                     1153
      R14 := S SHRL 8 + 1;                                                   1154
      IF R14 = 7 THEN MVC(0,TBUF(23),")");                                   1155
      R14 := SAVE14;                                                         1156
    END;  COMMENT  END OF CONVSTATE;                                         1157
                                                                             1158
PROCEDURE PRINTCFSM(R14);                                                    1159
  BEGIN INTEGER SAVE14;                                                      1160
    SAVE14 := R14;                                                           1161
    MVC(37,WBUF,"THE CFSM FOR THE GRAMMAR IS AS FOLLOWS");                   1162
    R0 := @WBUF; WRITE; MVC(37,WBUF,BLANK);  WRITE; WRITE;                   1163
    FOR R1 := 0 STEP 1 UNTIL NUMREADSTATES DO                                1164
    BEGIN                                                                    1165
      R2 := R1 SHLL 1; R3 := READTOLA(R2);                                   1166
      IF R3 ¬= R1 THEN                                                       1167
      BEGIN                                                                  1168
        MVC(10,WBUF,"READ STATE "); CVD(R1,CONWORK);                         1169
        UNPK(3,7,WBUF(13),CONWORK); SETZONE(WBUF(16));                       1170
        MVC(12,WBUF(18),"IS INADEQUATE");                                    1171
        MVC(37,WBUF(34),"THE FOLLOWING LOOK-AHEAD IS NECESSARY.");           1172
        WRITE; MVC(72,WBUF,BLANK); WRITE; WRITE; R4 := R3 SHRL 8;            1173
        WHILE R4 = LOOKAHEADORD OR R4 = LOOKAHEADEMPTY DO                    1174
        BEGIN                                                                1175
          S := R3; CONVSTATE; MVC(24,WBUF,TBUF);                             1176
          R3 := R3 AND #FF;                                                  1177
          MVC(0,WBUF(24),":"); MVC(11,WBUF(32),"LA SYM NUM =");              1178
          R4 := R4 - R4; IC(R4,LASYMNUM(R3)); R4 := R4 + 1;                  1179
          CVD(R4,CONWORK); UNPK(3,7,WBUF(45),CONWORK);                       1180
          SETZONE(WBUF(48)); MVC(0,WBUF(49),")");                            1181
          WRITE; MVC(49,WBUF,BLANK); WRITE;                                  1182
          MVC(8,WBUF,"LA SET OF"); R4 := R3 SHLL 1;                          1183
          R5 := SUCCSTATE(R4) AND #FF; R5 := R5 SHLL 1;                      1184
          R4 := PRODSTART(R5); R5 := R5 - R5; IC(R5,PRODARRAY(R4));          1185
          VPT := R5; FIND; R4 := 36 - CLENGTH;                               1186
          FOR R7 := 0 STEP 1 UNTIL CLENGTH DO                                1187
          BEGIN                                                              1188
            IC(R5,CBCD(R7)); STC(R5,WBUF(R4)); R4 := R4 + 1;                 1189
```

```
    END;                                                                    1190
    R4 := 39;                                                               1191
    FOR R7 := 0 STEP 1 UNTIL 1 DO                                           1192
    BEGIN                                                                   1193
      IC(R5,ARROW(R7)); STC(R5,WBUF(R4)); R4 := R4 + 1;                     1194
    END;                                                                    1195
    R4 := R3 SHLL 1; R5 := SUCCSTATE(R4);                                   1196
    S := R5; CONVSTATE; MVC(24,WBUF(42),TBUF);                              1197
    WRITE; MVC(67,WBUF,BLANK);                                              1198
    R3 := FAILSTATE(R4); S := R3; CONVSTATE;                                1199
    MVC(24,WBUF(42),TBUF);                                                  1200
    MVC(10,WBUF(30),"DEFAULT ->"); WRITE;                                   1201
    MVC(67,WBUF,BLANK); WRITE; WRITE;                                       1202
    R4 := R3 SHRL 8;                                                        1203
  END;                                                                      1204
END;                                                                        1205
R2 := R1 SHLL 1; R3 := LINEARTOARRAY(R2);                                   1206
S := R3; CONVSTATE; MVC(4,WBUF,"STATE");                                    1207
MVC(24,WBUF(6),TBUF);  MVC(0,WBUF(28),":");                                 1208
MVC(18,WBUF(33),"ACCESSING SYMBOL IS"); R4 := R4 - R4;                      1209
IC(R4,SYMBEFORE(R1)); VPT := R4; FIND; R4 := 54;                            1210
FOR R2 := 0 STEP 1 UNTIL CLENGTH DO                                         1211
BEGIN                                                                       1212
  IC(R5,CBCD(R2)); STC(R5,WBUF(R4)); R4 := R4 + 1;                          1213
END;                                                                        1214
IC(R5,")"); STC(R5,WBUF(R4)); WRITE;                                        1215
MVC(131,WBUF,BLANK); WRITE;                                                 1216
R2 := R1 SHLL 1; R3 := TREADSTART(R2); R7 := R7 - R7;                       1217
IC(R7,TRDNUM(R1)); R7 := R7 + R3 - 1;                                       1218
FOR R3 := R3 STEP 1 UNTIL R7 DO                                             1219
BEGIN                                                                       1220
  R2 := R2 - R2; IC(R2,TSYMLIST(R3)); VPT := R2; FIND;                      1221
  R4 := 36 - CLENGTH;                                                       1222
  FOR R6 := 0 STEP 1 UNTIL CLENGTH DO                                       1223
  BEGIN                                                                     1224
    IC(R5,CBCD(R6)); STC(R5,WBUF(R4)); R4 := R4 + 1;                        1225
  END;                                                                      1226
  R4 := 39;                                                                 1227
  FOR R6 := 0 STEP 1 UNTIL 1 DO                                             1228
  BEGIN                                                                     1229
    IC(R5,ARROW(R6)); STC(R5,WBUF(R4)); R4 := R4 + 1;                       1230
  END;                                                                      1231
  R12 := BASEREG008;                                                        1232
  R2 := R3 SHLL 1; R4 := TSTATELIST(R2); S := R4;                           1233
  R12 := BASE005;                                                           1234
  CONVSTATE; MVC(24,WBUF(42),TBUF);                                         1235
  WRITE; MVC(67,WBUF,BLANK);                                                1236
END;                                                                        1237
```

```
      MVC(46,WBUF(12),DASHLINE); WRITE; MVC(60,WBUF,BLANK);                1238
      R2 := R1 SHLL 1; R3 := NTREADSTART(R2); R7 := R7 - R7;               1239
      IC(R7,NTRDNUM(R1)); R7 := R7 + R3 - 1;                               1240
      FOR R3 := R3 STEP 1 UNTIL R7 DO                                      1241
      BEGIN                                                                1242
        R2 := R2 - R2; IC(R2,NTSYMLIST(R3)); VPT := R2; FIND;              1243
        R4 := 36 - CLENGTH;                                                1244
        FOR R6 := 0 STEP 1 UNTIL CLENGTH DO                                1245
        BEGIN                                                              1246
          IC(R5,CBCD(R6)); STC(R5,WBUF(R4)); R4 := R4 + 1;                 1247
        END;                                                               1248
        R4 := 39;                                                          1249
        FOR R6 := 0 STEP 1 UNTIL 1 DO                                      1250
        BEGIN                                                              1251
          IC(R5,ARROW(R6)); STC(R5,WBUF(R4)); R4 := R4 + 1;                1252
        END;                                                               1253
        R2 := R3 SHLL 1; R4 := NTSTATELIST(R2);                            1254
        S := R4; CONVSTATE; MVC(24,WBUF(42),TBUF); WRITE;                  1255
        MVC(67,WBUF,BLANK);                                                1256
      END;                                                                 1257
      WRITE; WRITE;                                                        1258
    END;                                                                   1259
    PAGE;                                                                  1260
    R14 := SAVE14;                                                         1261
  END;  COMMENT  END OF PRINTCFSM;                                         1262
                                                                           1263
    PROCEDURE ISVALIDLA(R14);                                              1264
      BEGIN ARRAY 3 INTEGER SAVEREGS;                                      1265
        STM(R1,R3,SAVEREGS);                                               1266
        R2 := SYM SHRL 5;                                                  1267
        R1 := NTSYM - NUMTERMINALS SHLL 3 OR R2 SHLL 2;                    1268
        R2 := SYM AND #1F; R3 := MASK SHRL R2;                             1269
        R2 := LOOKAHEADTABLE(R1) AND R3;                                   1270
        IF R2 = 0 THEN RESET(ISVALID) ELSE SET(ISVALID);                   1271
        LM(R1,R3,SAVEREGS);                                                1272
      END;  COMMENT  END OF ISVALIDLA;                                     1273
                                                                           1274
GLOBAL PROCEDURE LOOKAHEADANDDPDA(R14);                                    1275
  BEGIN INTEGER SAVE14;                                                    1276
    BYTE ERRORST;                                                          1277
                                                                           1278
    PROCEDURE VALIDATELAXITION(R14);                                       1279
      BEGIN ARRAY 4 INTEGER SAVEREGS;                                      1280
        STM(R1,R4,SAVEREGS);                                               1281
        R1 := NTSYM - NUMTERMINALS SHLL 3; R2 := SYM SHRL 5;               1282
        R1 := R1 OR R2 SHLL 2;                                             1283
        R2 := LOOKAHEADTABLE(R1); R4 := SYM AND #1F;                       1284
        R3 := MASK SHRL R4; R2 := R2 OR R3;                                1285
```

```
      LOOKAHEADTABLE(R1) := R2;                                          1286
      LM(R1,R4,SAVEREGS);                                                1287
    END;  COMMENT  END OF VALIDLAXITIIN;                                 1288
                                                                         1289
  PROCEDURE INTERSECT(R14);                                              1290
    BEGIN ARRAY 4 INTEGER SAVEREGS;                                      1291
      STM(R1,R4,SAVEREGS);                                               1292
      R1 := NTSYM - NUMTERMINALS SHLL 5;                                 1293
      R2 := NTSYM2 - NUMTERMINALS SHLL 5;                                1294
      RESET(INTER);                                                      1295
      FOR R3 := 0 STEP 1 UNTIL 3 DO                                      1296
      BEGIN                                                              1297
        R4 := LOOKAHEADTABLE(R1) AND LOOKAHEADTABLE(R2);                 1298
        IF R4 ¬= 0 THEN OI(#FF,INTER);                                   1299
        R1 := R1 + 4; R2 := R2 + 4;                                      1300
      END;                                                               1301
      LM(R1,R4,SAVEREGS);                                                1302
    END;  COMMENT  END OF INTERSECT;                                     1303
                                                                         1304
  PROCEDURE INCLUDED(R14);                                               1305
    BEGIN  ARRAY 5 INTEGER SAVEREGS;                                     1306
      STM(R1,R5,SAVEREGS);                                               1307
      R1 := SYM - NUMTERMINALS SHLL 5;                                   1308
      R2 := PRED - NUMTERMINALS SHLL 5;                                  1309
      SET(INCL);                                                         1310
      FOR R3 := 0 STEP 1 UNTIL 3 DO                                      1311
      BEGIN                                                              1312
        R4 := LOOKAHEADTABLE(R2);                                        1313
        R5 := LOOKAHEADTABLE(R1) AND R4;                                 1314
        IF R5 = R4 THEN NI(#FF,INCL)                                     1315
        ELSE NI(#00,INCL);                                               1316
        R2 := R2 + 4; R1 := R1 + 4;                                      1317
      END;                                                               1318
      LM(R1,R5,SAVEREGS);                                                1319
    END;  COMMENT  END OF INCLUDED;                                      1320
                                                                         1321
  PROCEDURE INCLUDE(R14);                                                1322
    BEGIN  ARRAY 4 INTEGER SAVEREGS;                                     1323
      STM(R1,R4,SAVEREGS);                                               1324
      R1 := SYM - NUMTERMINALS SHLL 5;                                   1325
      R2 := PRED - NUMTERMINALS SHLL 5;                                  1326
      FOR R3 := 0 STEP 1 UNTIL 3 DO                                      1327
      BEGIN                                                              1328
        R4 := LOOKAHEADTABLE(R1) OR LOOKAHEADTABLE(R2);                  1329
        LOOKAHEADTABLE(R1) := R4; R1 := R1 + 4;                          1330
        R2 := R2 + 4;                                                    1331
      END;                                                               1332
      LM(R1,R4,SAVEREGS);                                                1333
```

```
  END;  COMMENT  END OF INCLUDE;                                                1334
                                                                               1335
PROCEDURE NOTSLR1(R14);                                                        1336
  BEGIN INTEGER SAVE14;                                                        1337
    SAVE14 := R14;                                                             1338
    RESET(ISLR1); MVC(89,WBUF,STARLINE);                                       1339
    R0 := @WBUF; WRITE; MVC(90,WBUF,BLANK); WRITE; WRITE;                      1340
    CONVSTATE; MVC(25,WBUF,"THE GRAMMAR IS NOT SLR(1):");                      1341
    R14 := STATE; CVD(R14,CONWORK);                                            1342
    UNPK(3,7,WBUF(27),CONWORK); SETZONE(WBUF(30));                             1343
    MVC(23,WBUF(32),"INTERSECTS THAT OF STATE");                               1344
    MVC(24,WBUF(56),TBUF); MVC(0,WBUF(81),".");                                1345
    WRITE; MVC(82,WBUF,BLANK); WRITE; WRITE;                                   1346
    MVC(89,WBUF,STARLINE); WRITE; MVC(89,WBUF,BLANK); WRITE;                   1347
    R14 := SAVE14;                                                             1348
  END;  COMMENT  END OF NOTSLR1;                                               1349
                                                                               1350
PROCEDURE PRINTSLR1LASETS(R14);                                                1351
  BEGIN INTEGER SAVE14;                                                        1352
    SAVE14 := R14; R1 := NUMNTS; IF R1 > 18 THEN PAGE;                         1353
    R1 := NUMTERMINALS;                                                        1354
    IF R1 > 50 THEN                                                            1355
    BEGIN                                                                      1356
      WRITE; MVC(17,WBUF,"NUMTERMINALS > 50,"); WRITE;                         1357
      MVC(33,WBUF,"ONLY THE FIRST 50 WILL BE PRINTED.");                       1358
      WRITE; MVC(33,WBUF,BLANK); WRITE;                                        1359
    END;                                                                       1360
    MVC(36,WBUF(11),"T H E   L G O K - A H E A D   S E T S");                  1361
    R0 := @WBUF; WRITE; MVC(36,WBUF(11),BLANK); WRITE; WRITE;                  1362
    MVC(31,WBUF,"NONTERMINAL     TERMINAL SYMBOLS");                           1363
    WRITE; MVC(31,WBUF,BLANK);                                                 1364
    MVC(5,WBUF(3),"SYMBOL"); R2 := 11;                                         1365
    FOR R4 := 1 STEP 1 UNTIL 2 DO                                              1366
    BEGIN                                                                      1367
      FOR R1 := R4 STEP 2 UNTIL NUMTERMINALS DO                                1368
      BEGIN                                                                    1369
        IF R1 < 10 THEN                                                        1370
        BEGIN                                                                  1371
          R2 := R2 + 4; CVD(R1,CONWORK); R3 := @WBUF(R2);                      1372
          UNPK(0,7,B3,CONWORK); SETZONE(B3);                                   1373
        END ELSE                                                               1374
        BEGIN                                                                  1375
          R2 := R2 + 4; CVD(R1,CONWORK); R3 := @WBUF(R2)- 1;                   1376
          UNPK(1,7,B3,CONWORK); R3 := R3 + 1;                                  1377
          SETZONE(B3);                                                         1378
        END;                                                                   1379
      END;                                                                     1380
      R0 := @WBUF; WRITE; MVC(131,WBUF,BLANK);                                 1381
```

```
      R2 := 13;                                                          1382
    END;                                                                 1383
    IC(R1,DASHLINE); R2 := NUMTERMINALS SHLL 1 + 13;                     1384
    FOR R3 := 13 STEP 1 UNTIL R2 DO STC(R1,WBUF(R3));                    1385
    WRITE; MVC(131,WBUF,BLANK);                                          1386
    R2 := NUMTERMINALS + NUMNTS;                                         1387
    FOR R1 := NUMTERMINALS + 1 STEP 1 UNTIL R2 DO                        1388
    BEGIN                                                                1389
      VPT := R1; FIND; R3 := 11 - CLENGTH;                               1390
      IF R3 < 0 THEN                                                     1391
      BEGIN                                                              1392
        R4 := CLENGTH + R3; R3 := 0;                                     1393
      END ELSE R4 := CLENGTH;                                            1394
      FOR R5 := 0 STEP 1 UNTIL R4 DO                                     1395
      BEGIN                                                              1396
        IC(R6,CBCD(R5)); STC(R6,WBUF(R3)); R3 := R3 + 1;                 1397
      END;                                                               1398
      MVC(0,WBUF(13),"|"); R3 := 15; R4 := #F1;                          1399
      R7 := NUMTERMINALS; IF R7 > 50 THEN R7 := 50;                      1400
      FOR R5 := 1 STEP 1 UNTIL R7 DO                                     1401
      BEGIN                                                              1402
        NTSYM := R1; SYM := R5; ISVALIDLA;                               1403
        IF ISVALID THEN STC(R4,WBUF(R3)); R3 := R3 + 2;                  1404
      END;                                                               1405
      WRITE; MVC(131,WBUF,BLANK);                                        1406
    END;                                                                 1407
    PAGE; R14 := SAVE14;                                                 1408
  END;  COMMENT  END OF PRINTSLR1LASETS;                                 1409
                                                                         1410
                                                                         1411
R8 := BASE011; SAVE14 := R14;                                            1412
FOR R1 := 1 STEP 1 UNTIL NUMREADSTATES DO                                1413
BEGIN                                                                    1414
  R8 := BASE010; R2 := R2 - R2; IC(R2,SYMBEFORE(R1));                    1415
  R8 := BASE011; NTSYM := R2;                                            1416
  R3 := @ONLEFT(R2);                                                     1417
  CLC(0,B3,TRUE);                                                        1418
  IF = THEN                                                              1419
  BEGIN                                                                  1420
    R2 := R1 SHLL 1; R3 := TREADSTART(R2);                               1421
    R2 := R2 - R2; IC(R2,TRDNUM(R1)); R2 := R2 + R3 SHLL 1;              1422
    FOR R3 := R3 SHLL 1 STEP 2 UNTIL R2 DO                               1423
    BEGIN                                                                1424
      R12 := BASEREG008; R4 := TSTATELIST(R3); R12 := BASE005;           1425
      IF R4 ¬= #6FF THEN                                                 1426
      BEGIN                                                              1427
        R4 := R3 SHRL 1; R5 := R5 - R5;                                  1428
        IC(R5,TSYMLIST(R4)); SYM := R5; VALIDATELAXITION;                1429
```

```
      END;                                                              1430
    END;                                                                1431
  END;                                                                  1432
END;                                                                    1433
R2 := NUMPRODS - 1;                                                     1434
FOR R1 := 1 STEP 1 UNTIL R2 DO                                          1435
BEGIN                                                                   1436
  R3 := R3 - R3; IC(R3,RTPTSIZE(R1));                                   1437
  IF R3 ¬= 0 THEN                                                       1438
  BEGIN                                                                 1439
    R4 := R1 SHLL 1; R5 := PRODSTART(R4); R5 := R5 + R3;                1440
    R4 := R4 - R4; IC(R4,PRODARRAY(R5)); R3 := @ONLEFT(R4);             1441
    CLC(0,B3,TRUE);                                                     1442
    IF = THEN                                                           1443
    BEGIN                                                               1444
      NTSYM := R4; R1 := R1 SHLL 1; R4 := PRODSTART(R1);                1445
      R5 := R5 - R5; IC(R5,PRODARRAY(R4)); SYM := R5;                   1446
      VALIDATELAXITION; R1 := R1 SHRL 1;                                1447
    END;                                                                1448
  END;                                                                  1449
END;                                                                    1450
SET(CHANGING);                                                          1451
WHILE CHANGING DO                                                       1452
BEGIN                                                                   1453
  RESET(CHANGING); R2 := NUMTERMINALS + NUMNTS;                         1454
  FOR R1 := NUMTERMINALS + 1 STEP 1 UNTIL R2 DO                         1455
  BEGIN                                                                 1456
    FOR R3 := NUMTERMINALS + 1 STEP 1 UNTIL R2 DO                       1457
    BEGIN                                                               1458
      NTSYM := R1; SYM := R3; ISVALIDLA;                                1459
      IF ISVALID THEN                                                   1460
      BEGIN                                                             1461
        SYM := R1; PRED := R3; INCLUDED;                                1462
        IF ¬INCL THEN                                                   1463
        BEGIN                                                           1464
          SET(CHANGING); INCLUDE;                                       1465
        END;                                                            1466
      END;                                                              1467
    END;                                                                1468
  END;                                                                  1469
END;                                                                    1470
R2 := NUMLASTATES - 1;                                                  1471
FOR R1 := 0 STEP 1 UNTIL R2 DO                                          1472
BEGIN                                                                   1473
  STATE := R1; R3 := R1 SHLL 1; R4 := FAILSTATE(R3); S:= R4;            1474
  R5 := SUCCSTATE(R3) AND #FF SHLL 1;                                   1475
  R6 := PRODSTART(R5); R7 := R7 - R7; IC(R7,PRODARRAY(R6));             1476
  NTSYM := R7; R3 := R4 AND #FF SHLL 1;                                 1477
```

```
      R5 := SUCCSTATE(R3) AND #FF SHLL 1;                                    1478
      R6 := PRODSTART(R5); R7 := R7 - R7; IC(R7,PRODARRAY(R6));              1479
      NTSYM2 := R7; INTERSECT;                                               1480
      WHILE R4 > 767 AND ¬INTER DO                                           1481
      BEGIN                                                                  1482
        R3 := R4 AND #FF SHLL 1;  R4 := FAILSTATE(R3);                       1483
        R3 := R4 AND #FF SHLL 1; S := R4;                                    1484
        R5 := SUCCSTATE(R3) AND #FF SHLL 1; R3 := PRODSTART(R5);             1485
        R5 := R5 - R5; IC(R5,PRODARRAY(R3)); NTSYM2 := R5;                   1486
        INTERSECT;                                                           1487
      END;                                                                   1488
      R3 := R4 SHRL 8;                                                       1489
      IF R3 = LOOKAHEADORD OR R3 = LOOKAHEADEMPTY THEN                       1490
      BEGIN                                                                  1491
        NOTSLR1;                                                             1492
      END ELSE                                                               1493
      BEGIN                                                                  1494
        R3 := R4 AND #FF SHLL 1; R5 := TREADSTART(R3);                       1495
        R6 := R6 - R6; IC(R6,TSYMLIST(R5)); R7 := R7 - R7;                   1496
        R3 := S AND #FF;                                                     1497
        IC(R7,TRDNUM(R3)); R7 := R7 + R5;                                    1498
        R12 := BASEREG008; R3 := R5 SHLL 1; R4 := TSTATELIST(R3);            1499
        R12 := BASE005; SYM := R6; ISVALIDLA;                                1500
        IF R4 = #6FF OR ¬ISVALID THEN SET(ERRORST)                           1501
        ELSE RESET(ERRORST);                                                 1502
        WHILE R5 < R7 AND ERRORST DO                                         1503
        BEGIN                                                                1504
          R5 := R5 + 1; R6 := R6 - R6; IC(R6,TSYMLIST(R5));                  1505
          R12 := BASEREG008; R3 := R5 SHLL 1; R4 := TSTATELIST(R3);          1506
          R12 := BASE005; SYM := R6; ISVALIDLA;                              1507
          IF R4 = #6FF OR ¬ISVALID THEN SET(ERRORST)                         1508
          ELSE RESET(ERRORST);                                               1509
        END;                                                                 1510
        IF R5 < R7 THEN NOTSLR1;                                             1511
      END;                                                                   1512
    END;                                                                     1513
    MVC(34,WBUF,"LOOK-AHEAD SETS HAVE BEEN COMPUTED.");                      1514
    WRITE; MVC(34,WBUF,BLANK);                                               1515
    IF ISLR1 THEN                                                            1516
    BEGIN                                                                    1517
      MVC(20,WBUF,"THE GRAMMAR IS LR(1).");                                  1518
      WRITE; MVC(20,WBUF,BLANK);                                             1519
    END;                                                                     1520
    WRITE; WRITE;                                                            1521
    R1 := ERRCOUNT;                                                          1522
    IF LASETSLIST OR ¬ISLR1 OR R1 ¬= 0 THEN PRINTSLR1LASETS;                 1523
    R14 := SAVE14; R12 := BASE010;                                           1524
  END;  COMMENT  END OF LOOKAHEADAND DPDA;                                   1525
```

```
GLOBAL PROCEDURE COMPUTEDPDA(R14);
  BEGIN INTEGER SAVE14;

    ARRAY 255 SHORT INTEGER REDSUCCFORNT SYN NEWV(500);
    ARRAY 255 SHORT INTEGER RESUMELIST SYN NEWV(0);
    ARRAY 255 BYTE FREGLIST SYN BSSIZE(0);
    ARRAY 255 SHORT INTEGER LBLIST SYN BASISSTACK(0);

    INTEGER LISTPTR, LBST, RESSTATE, TEMP, LBPTR;
    BYTE NOTFOUND, NEWLBSTATE;
    SHORT INTEGER MN, MNPTR, MX, MXPTR;


    PROCEDURE ENTER(R14);
      BEGIN INTEGER SAVE14; ARRAY 3 INTEGER SAVEREGS;
        SAVE14 := R14; STM(R1,R3,SAVEREGS); R1 := 0;
        IF R1 < LISTPTR THEN SET(NOTFOUND)
        ELSE RESET(NOTFOUND);
        WHILE NOTFOUND DO
        BEGIN
          R2 := R1 SHLL 1; R3 := RESUMELIST(R2);
          IF R3 = RESSTATE THEN
          BEGIN
            IC(R2,FREGLIST(R1)); R2 := R2 + 1;
            STC(R2,FREGLIST(R1)); RESET(NOTFOUND);
          END ELSE
          BEGIN
            R1 := R1 + 1;
            IF R1 < LISTPTR THEN SET(NOTFOUND)
            ELSE RESET(NOTFOUND);
          END;
        END;
          R2 := LISTPTR;
          IF R1 = LISTPTR THEN
          BEGIN
            R3 := 1; STC(R3,FREGLIST(R2));
          END;
          R2 := LISTPTR SHLL 1;
        R3 := LBST; STH(R3,LBLIST(R2)); R3 := RESSTATE;
        STH(R3,RESUMELIST(R2)); R2 := LISTPTR + 1; LISTPTR := R2;
        R14 := SAVE14; LM(R1,R3,SAVEREGS);
      END;  COMMENT  END OF ENTER;

    PROCEDURE INCLBPTR(R14);
      BEGIN INTEGER SAVE14;
        SAVE14 := R14; R14 := LBPTR + 1;
        IF R14 < 256 THEN LBPTR := R14
```

```
    ELSE BEGIN                                                          1574
      MVC(29,WBUF(15),"TOO MANY LOOK BACK TRANSITION");                 1575
      ERROR;                                                            1576
    END;                                                                1577
    R14 := SAVE14;                                                      1578
  END;  COMMENT  END OF INCLBPTR;                                       1579
                                                                        1580
PROCEDURE INCNUMLBSTATES(R14);                                          1581
  BEGIN INTEGER SAVE14;                                                 1582
    SAVE14 := R14; R14 := NUMLBSTATES + 1;                              1583
    IF R14 < 128 THEN NUMLBSTATES := R14                                1584
    ELSE BEGIN                                                          1585
      MVC(24,WBUF(15),"TOO MANY LOOK BACK STATES");                     1586
      ERROR;                                                            1587
    END;                                                                1588
    R14 := SAVE14;                                                      1589
  END;  COMMENT  END OF INCNUMLBSTATES;                                 1590
                                                                        1591
PROCEDURE ENCODELB(R14);                                                1592
  BEGIN INTEGER SAVE14; ARRAY 2 INTEGER SAVEREGS;                       1593
    SAVE14 := R14; STM(R1,R2,SAVEREGS);                                 1594
    R1 := 255; MN := R1; MNPTR := R1;                                   1595
    R1 := 0; MX := R1; MXPTR := R1; R7 := LISTPTR - 1;                  1596
    FOR R1 := 0 STEP 1 UNTIL R7 DO                                      1597
    BEGIN                                                               1598
      R2 := R2 - R2; IC(R2,FREGLIST(R1));                               1599
      IF R2 < MN AND R2 ¬= 0 THEN                                       1600
      BEGIN                                                             1601
        MN := R2; MNPTR := R1;                                          1602
      END;                                                              1603
      IF R2 >= MX THEN                                                  1604
      BEGIN                                                             1605
        MX := R2; MXPTR := R1;                                          1606
      END;                                                              1607
    END;                                                                1608
    RESET(NEWLBSTATE); R1 := MNPTR;                                     1609
    IF R1 ¬= MXPTR THEN                                                 1610
    BEGIN                                                               1611
      SET(NEWLBSTATE); R1 := NUMLBSTATES; R2 := LBPTR;                  1612
      STC(R2,LBSTART(R1)); R2 := 0; STC(R2,LBNUM(R1));                  1613
    END;                                                                1614
    R1 := MNPTR;                                                        1615
    WHILE R1 ¬= MXPTR DO                                                1616
    BEGIN                                                               1617
      R2 := 0; STC(R2,FREGLIST(R1)); R2 := NUMLBSTATES;                 1618
      R3 := R3 - R3; IC(R3,LBNUM(R2)); R3 := R3 + MN;                   1619
      STC(R3,LBNUM(R2));  R2 := LISTPTR - 1 SHLL 1;                     1620
      FOR R3 := 0 STEP 2 UNTIL R2 DO                                    1621
```

```
      BEGIN                                                                 1622
        R4 := MNPTR SHLL 1; R5 := RESUMELIST(R3);                           1623
        IF R5 = RESUMELIST(R4) THEN                                         1624
        BEGIN                                                               1625
          R4 := LBPTR SHLL 1; R5 := RESUMELIST(R3);                         1626
          RESUMESTATE(R4) := R5; R5 := LBLIST(R3);                          1627
          LBSTATE(R4) := R5; INCLBPTR;                                      1628
        END;                                                                1629
      END;                                                                  1630
      R2 := 255; MN := R2; R3 := LISTPTR - 1;                               1631
      FOR R4 := 0 STEP 1 UNTIL R3 DO                                        1632
      BEGIN                                                                 1633
        R2 := R2 - R2; IC(R2,FREGLIST(R4));                                 1634
        IF R2 < MN AND R2 ¬= 0 THEN                                         1635
        BEGIN                                                               1636
          MN := R2; MNPTR := R4; R1 := R4;                                  1637
        END;                                                                1638
      END;                                                                  1639
    END;                                                                    1640
    IF NEWLBSTATE THEN                                                      1641
    BEGIN                                                                   1642
      R1 := MXPTR SHLL 1; R2 := LBPTR SHLL 1;                               1643
      R3 := RESUMELIST(R1); RESUMESTATE(R2) := R3; INCLBPTR;                1644
      R1 := LOOKBACK SHLL 8 OR NUMLBSTATES;                                 1645
      R2 := SYM SHLL 1; REDSUCCFORNT(R2) := R1;                             1646
      INCNUMLBSTATE;                                                        1647
    END ELSE                                                                1648
    BEGIN                                                                   1649
      R1 := RESUMELIST(0); R2 := SYM SHLL 1;                                1650
      REDSUCCFORNT(R2) := R1;                                               1651
    END;                                                                    1652
    R14 := SAVE14; LM(R1,R2,SAVEREGS);                                      1653
  END;  COMMENT  END OF ENCODELB;                                           1654
                                                                            1655
PROCEDURE PRINTDPDA(R14);                                                   1656
  BEGIN INTEGER SAVE14;                                                     1657
    SAVE14 := R14;                                                          1658
    MVC(39,WBUF,"THE DPDA FOR THE GRAMMAR CONSISTS OF THE");                1659
    MVC(36,WBUF(41),"TERMINAL TRANSITIONS OF THE CFSM PLUS");               1660
    R0 := @WBUF; WRITE; MVC(80,WBUF,BLANK);                                 1661
    MVC(33,WBUF,"THE FOLLOWING REDUCE AND LOOK-BACK");                      1662
    MVC(10,WBUF(35),"TRANSITIONS");                                         1663
    WRITE; MVC(50,WBUF,BLANK); WRITE; WRITE;                                1664
    FOR R1 := 1 STEP 1 UNTIL NUMPRODS DO                                    1665
    BEGIN                                                                   1666
      R2 := R1 SHLL 1; R3 := REDUCESUCC(R2);                                1667
      S:= R3; CONVSTATE; MVC(11,WBUF,"STATE REDUCE");                       1668
      R2 := 13; CVD(R1,CONWORK);                                            1669
```

```
      IF R1 < 10 THEN UNPK(0,7,WBUF(13),CONWORK)                          1670
      ELSE BEGIN                                                          1671
        UNPK(1,7,WBUF(13),CONWORK); R2 := 14;                             1672
      END;                                                                1673
    R3 := @WBUF(R2); SETZONE(B3); R3 := R3 + 1;                           1674
      MVC(0,B3,":"); MVC(2,WBUF(21),"POP"); R3 := R3 - R3;                1675
      R12 := BASE005;                                                     1676
      IC(R3,RTPTSIZE(R1)); R3 := R3 - 1; CVD(R3,CONWORK);                 1677
      R12 := BASEREG008;                                                  1678
      UNPK(1,7,WBUF(25),CONWORK); SETZONE(WBUF(26));                      1679
      MVC(1,WBUF(38),ARROW); MVC(24,WBUF(42),TBUF);                       1680
      WRITE; MVC(70,WBUF,BLANK); WRITE;                                   1681
    END;                                                                  1682
    WRITE;                                                                1683
    R2 := NUMLBSTATES - 1;                                                1684
    FOR R1 := 0 STEP 1 UNTIL R2 DO                                        1685
    BEGIN                                                                 1686
      MVC(14,WBUF,"STATE LOOK BACK"); CVD(R1,CONWORK);                    1687
      UNPK(1,7,WBUF(16),CONWORK);                                         1688
      SETZONE(WBUF(17)); MVC(0,WBUF(18),":"); WRITE;                      1689
      MVC(18,WBUF,BLANK); R3 := R3 - R3; IC(R3,LBSTART(R1));              1690
      R4 := R4 - R4; IC(R4,LBNUM(R1));                                    1691
      R7 := R3 + R4 - 1;                                                  1692
      FOR R3 := R3 STEP 1 UNTIL R7 DO                                     1693
      BEGIN                                                               1694
        R4 := R3 SHLL 1; R5 := LBSTATE(R4); S := R5; CONVSTATE;           1695
        MVC(24,WBUF(16),TBUF); MVC(1,WBUF(38),ARROW);                     1696
        R5 := RESUMESTATE(R4); S := R5; CONVSTATE;                        1697
        MVC(24,WBUF(42),TBUF); WRITE; MVC(70,WBUF,BLANK);                 1698
      END;                                                                1699
      R4 := R3 SHLL 1; R5 := RESUMESTATE(R4);                             1700
      S := R5; CONVSTATE; MVC(6,WBUF(16),"DEFAULT");                      1701
      MVC(1,WBUF(38),ARROW); MVC(24,WBUF(42),TBUF);                       1702
      WRITE; MVC(70,WBUF,BLANK); WRITE; WRITE;                            1703
    END;                                                                  1704
    R14 := SAVE14;                                                        1705
  END;  COMMENT  END OF PRINT DPDA;                                       1706
                                                                          1707
                                                                          1708
SAVE14 := R14; R8 := BASE010; R12 := BASEREG008;                          1709
R1 := 0; NUMLBSTATES := R1; LBPTR := R1;                                  1710
R1 := NUMREADSTATES + 1; LISTPTR := R1;                                   1711
R2 := NUMTERMINALS + NUMNTS; TEMP := R2;                                  1712
FOR R1 := NUMTERMINALS + 1 STEP 1 UNTIL TEMP DO                           1713
BEGIN                                                                     1714
  SYM := R1; R2 := 0; R4 := LISTPTR - 1;                                  1715
  FOR R3 := 0 STEP 1 UNTIL R4 DO STC(R2,FREGLIST(R3));                    1716
  LISTPTR := R2; R7 := NUMREADSTATES SHLL 1;                              1717
```

```
      FOR R3 := 0 STEP 2 UNTIL R7 DO                                          1718
      BEGIN                                                                   1719
        R2 := R2 - R2; R4 := R3 SHRL 1; IC(R2,NTRDNUM(R4));                   1720
        R2 := R2 + NTREADSTART(R3) - 1;                                       1721
        FOR R4 := NTREADSTART(R3) STEP 1 UNTIL R2 DO                          1722
        BEGIN                                                                 1723
          R5 := R5 - R5; IC(R5,NTSYMLIST(R4));                                1724
          IF R5 = SYM THEN                                                    1725
          BEGIN                                                               1726
            R4 := R4 SHLL 1;                                                  1727
            R5 := LINEARTOARRAY(R3); R6 := NTSTATELIST(R4);                   1728
            R4 := R4 SHRL 1;                                                  1729
            LBST := R5; RESSTATE := R6; ENTER;                                1730
            R5 := R5 - R5; R6 := R3 SHRL 1; IC(R5,NTRDNUM(R6));               1731
            R4 := R4 + R5;                                                    1732
          END;                                                                1733
        END;                                                                  1734
      END;                                                                    1735
      ENCODELB;                                                               1736
    END;                                                                      1737
    R2 := NUMREDUCESTATES - 1 SHLL 1;                                         1738
    FOR R1 := 0 STEP 2 UNTIL R2 DO                                            1739
    BEGIN                                                                     1740
      R3 := PRODSTART(R1); R4 := R4 - R4; R12 := BASEQ05;                     1741
      IC(R4,PRODARRAY(R3)); R12 := BASEREG008; R4 := R4 SHLL 1;               1742
      R5 := REDSUCCFORNT(R4); REDUCESUCC(R1) := R5;                           1743
    END;                                                                      1744
    R1 := R1 + 2; R2 := EXIT SHLL 8; REDUCESUCC(R1) := R2;                    1745
    MVC(26,WBUF,"THE DPDA HAS BEEN COMPUTED,"); WRITE;                        1746
    MVC(26,WBUF,BLANK); WRITE;                                                1747
    IF DPDALIST THEN PRINTDPDA;                                               1748
    R14 := SAVE14;                                                            1749
  END;  COMMENT  END OF COMPUTEDPDA;                                          1750
                                                                              1751
GLOBAL PROCEDURE PUNCHDPDA(R14);                                              1752
  BEGIN                                                                       1753
    INTEGER SAVE14, NUMLINE, HEXLENGTH, DECLENGTH, NUMBITS, CP;               1754
    INTEGER ADDRESS, ARRAYLENGTH, TEMP;                                       1755
    ARRAY 16 BYTE HEXCODE = ("0123456789ABCDEF");                             1756
    ARRAY 10 BYTE HEXLINE = "#";                                              1757
    BYTE ARRAYEND, LL;                                                        1758
    ARRAY 18 INTEGER SEGTABLE;                                                1759
    INTEGER BYTECNT = 0;                                                      1760
    INTEGER SEGPT = 0;                                                        1761
                                                                              1762
                                                                              1763
    PROCEDURE PUNCHCOMMENT(R14);                                              1764
      BEGIN INTEGER SAVE14;                                                   1765
```

```
            SAVE14 := R14; CVD(R1,CONWORK);                                    1766
            R3 := @WBUF(R2); UNPK(3,7,B3,CONWORK); R3 := R3 + 3;               1767
            SETZONE(B3);   WRITE; PUNCH; MVC(80,WBUF,BLANK);                   1768
            WRITE; PUNCH; R14 := SAVE14;                                       1769
          END;  COMMENT  END OF PUNCHCOMMENT;                                  1770
                                                                               1771
        PROCEDURE PUNCHDECLARE(R14);                                           1772
          BEGIN INTEGER SAVE14; SAVE14 := R14;                                 1773
            R1 := ARRAYLENGTH + 1;                                             1774
            MVC(4,WBUF(2),"ARRAY"); CVD(R1,CONWORK);                           1775
            UNPK(3,7,WBUF(8),CONWORK); SETZONE(WBUF(11));                      1776
            R14 := NUMBITS;                                                    1777
            IF LL THEN                                                         1778
            BEGIN                                                              1779
              MVC(6,WBUF(13),"INTEGER"); R1 := 21; GOTO JUMP;                  1780
            END;                                                               1781
            IF R14 = 32 THEN                                                   1782
            BEGIN                                                              1783
              MVC(3,WBUF(13),"REAL"); R1 := 18; GOTO JUMP;                     1784
            END;                                                               1785
            IF R14 = 8 THEN                                                    1786
            BEGIN                                                              1787
              MVC(3,WBUF(13),"BYTE"); R1 := 18;                                1788
            END ELSE                                                           1789
            BEGIN                                                              1790
              MVC(12,WBUF(13),"SHORT INTEGER"); R1 := 27;                      1791
            END;                                                               1792
JUMP:       FOR R2 := 0 STEP 1 UNTIL DECLENGTH DO                              1793
            BEGIN                                                              1794
              IC(R3,TBUF(R2)); STC(R3,WBUF(R1)); R1 := R1 + 1;                 1795
            END;                                                               1796
            R1 := R1 + 1; R2 := @WBUF(R1);                                     1797
            MVC(2,B2,"= ("); R1 := R1 + 3; CP := R1;                           1798
            R14 := SAVE14;                                                     1799
          END;, COMMENT  END OF PUNCHDECLARE;                                  1800
                                                                               1801
        PROCEDURE BUILDHEXLINE(R14);                                           1802
          BEGIN INTEGER SAVE14; ARRAY 5 INTEGER SAVEREGS;                      1803
            SAVE14 := R14; STM(R1,R5,SAVEREGS);                                1804
            R14 := NUMLINE;                                                    1805
            R3 := NUMBITS - 4;                                                 1806
            R2 := NUMBITS SHRL 2;                                              1807
            FOR R1 := 0 STEP 4 UNTIL R3 DO                                     1808
            BEGIN                                                              1809
              R5 := NUMLINE SHRL R1 AND #F;                                    1810
              IC(R4,HEXCODE(R5)); STC(R4,HEXLINE(R2)); R2 := R2 - 1;           1811
            END;                                                               1812
            R1 := NUMBITS;                                                     1813
```

```
      IF R1 = 16 THEN MVC(0,HEXLINE(5),"S");                                1814
      IF R1 = 8 THEN MVC(0,HEXLINE(3),"X");                                 1815
      R14 := SAVE14; LM(R1,R5,SAVEREGS);                                    1816
    END;  COMMENT  END OF BUILDHEXLINE;                                     1817
                                                                            1818
PROCEDURE PUNCHCARD(R14);                                                   1819
  BEGIN INTEGER SAVE14; SAVE14 := R14;                                      1820
    R14 := CP + HEXLENGTH;                                                  1821
    IF R14 >= 70 THEN                                                       1822
    BEGIN                                                                   1823
      R0 := @WBUF; WRITE; PUNCH; MVC(72,WBUF,BLANK); R14 := 10;             1824
    END ELSE R14 := CP;                                                     1825
    FOR R7 := 0 STEP 1 UNTIL HEXLENGTH DO                                   1826
    BEGIN                                                                   1827
      IC(R6,HEXLINE(R7)); STC(R6,WBUF(R14)); R14 := R14 + 1;                1828
    END;                                                                    1829
    IF ARRAYEND THEN                                                        1830
    BEGIN                                                                   1831
      IC(R7,")"); STC(R7,WBUF(R14)); R14 := R14 + 1;                        1832
      IC(R7,";"); STC(R7,WBUF(R14));  WRITE; PUNCH;                         1833
      MVC(72,WBUF,BLANK); R14 := 10;                                        1834
    END ELSE                                                                1835
    BEGIN                                                                   1836
      IC(R7,","); STC(R7,WBUF(R14));R14 := R14 + 2;                         1837
    END;                                                                    1838
    CP := R14; R14 := SAVE14;                                               1839
  END;  COMMENT  OF PUNCHCARD;                                              1840
                                                                            1841
PROCEDURE PUNCHARRAY(R14);                                                  1842
  BEGIN INTEGER SAVE14;                                                     1843
    SAVE14 := R14;                                                          1844
    PUNCHDECLARE;                                                           1845
    R1 := ADDRESS; RESET(ARRAYEND);                                         1846
    R2 := NUMBITS;                                                          1847
    IF R2 = 32 THEN                                                         1848
    BEGIN                                                                   1849
      R2 := 8; HEXLENGTH := R2;                                             1850
      R2 := ARRAYLENGTH SHLL 2;                                             1851
      FOR R3 := 0 STEP 4 UNTIL R2 DO                                        1852
      BEGIN                                                                 1853
        LOAD(R4,B1); NUMLINE := R4;                                         1854
        IF R3 = R2 THEN SET(ARRAYEND) ELSE RESET(ARRAYEND);                 1855
        BUILDHEXLINE; PUNCHCARD; R1 := R1 + 4;                              1856
      END;                                                                  1857
      R2 := ARRAYLENGTH + 1 SHLL 2;                                         1858
      GOTO ENDARRAY;                                                        1859
    END;                                                                    1860
    IF R2 = 8 THEN                                                          1861
```

```
          BEGIN                                                          1862
            R2 := 3; HEXLENGTH := R2;                                    1863
            FOR R3 := 0 STEP 1 UNTIL ARRAYLENGTH DO                      1864
            BEGIN                                                        1865
              R2 := R2 - R2; IC(R2,B1); NUMLINE := R2;                   1866
              IF R3 = ARRAYLENGTH THEN SET(ARRAYEND);                    1867
              BUILDHEXLINE; PUNCHCARD; R1 := R1 + 1;                     1868
            END;                                                         1869
            R2 := ARRAYLENGTH + 1;                                       1870
          END ELSE                                                       1871
          BEGIN                                                          1872
            R2 := 5; HEXLENGTH := R2;                                    1873
            R2 := ARRAYLENGTH SHLL 1;                                    1874
            FOR R3 := 0 STEP 2 UNTIL R2 DO                               1875
            BEGIN                                                        1876
              LH(R4,B1); NUMLINE:= R4;                                   1877
              IF R3 = R2 THEN SET(ARRAYEND);                             1878
              BUILDHEXLINE; PUNCHCARD; R1 := R1 + 2;                     1879
            END;                                                         1880
            R2 := ARRAYLENGTH + 1 SHLL 1;                                1881
          END;                                                           1882
ENDARRAY:                                                                1883
          R1 := NUMBITS; R3 := BYTECNT;                                  1884
          IF R1 > 8 THEN                                                 1885
          BEGIN                                                          1886
            IF R1 = 16 THEN                                              1887
            BEGIN                                                        1888
              R4 := R3 AND #1;                                           1889
              IF R4 > 0 THEN R4 := R3 + 1                                1890
              ELSE R4 := R3;                                             1891
            END ELSE                                                     1892
            BEGIN                                                        1893
              R4 := R3 AND #3;                                           1894
              IF R4 > 0 THEN R4 := R3 AND #FFFFFFFC + 4                  1895
              ELSE R4 := R3;                                             1896
            END;                                                         1897
          END ELSE R4 := R3;                                             1898
          R3 := SEGPT SHLL 2; SEGTABLE(R3) := R4;                        1899
          R3 := R3 SHRL 2 + 1; SEGPT := R3;                              1900
          R4 := R4 + R2; BYTECNT := R4;                                  1901
          WRITE; PUNCH;                                                  1902
          R14 := SAVE14;                                                 1903
        END;  COMMENT  END OF PUNCHARRAY;                                1904
                                                                         1905
      PROCEDURE PUNCHLASETS(R14);                                        1906
        BEGIN INTEGER SAVE14;                                            1907
          SAVE14 := R14; R12 := BASE005; R8 := BASE011;                  1908
          RESET(ARRAYEND);                                               1909
```

```
R4 := BYTECNT;                                                          1910
R3 := SEGPT SHLL 2; SEGTABLE(R3) := R4;                                 1911
BYTECNT := R4; R3 := R3 SHRL 2 + 1; SEGPT := R3;                        1912
R1 := 8;  NUMBITS := R1; MVC(6,TBUF,"LATABLE");                         1913
R1 := 6; DECLENGTH := R1;                                               1914
R1 := NUMTERMINALS + 1 SHRL 3;                                          1915
R2 := NUMTERMINALS + 1 AND #7;                                          1916
IF R2 > 0 THEN R1 := R1 + 1;                                            1917
R1 := R1 * NUMLASTATES - 1; ARRAYLENGTH := R1;                          1918
PUNCHDECLARE;                                                           1919
R0 := @WBUF; WRITE; PUNCH; MVC(28,WBUF,BLANK);                          1920
R14 := NUMLASTATES - 1; TEMP := R14;                                    1921
FOR R1 := 0 STEP 1 UNTIL TEMP DO                                        1922
BEGIN                                                                   1923
  R2 := R1 SHLL 1; R3 := SUCCSTATE(R2) AND #FF SHLL 1;                  1924
  R2 := PRODSTART(R3); R3 := R3 - R3; IC(R3,PRODARRAY(R2));             1925
  NTSYM := R3; R7 := 0;                                                 1926
  R2 := NTSYM; VPT := R2; FIND; R3 := 19;                               1927
  FOR R2 := 0 STEP 1 UNTIL CLENGTH DO                                   1928
  BEGIN                                                                 1929
    IC(R4,CBCD(R2)); STC(R4,WBUF(R3)); R3 := R3 + 1;                    1930
  END;                                                                  1931
  MVC(6,WBUF(10),"COMMENT"); IC(R4,";"); STC(R4,WBUF(R3));              1932
  R0 := @WBUF; WRITE; PUNCH; MVC(80,WBUF,BLANK);                        1933
  R7 := 8; NUMBITS := R7; R6 := 3;                                      1934
  HEXLENGTH := R6; R6 := 10; CP := R6; R6 := 1;                         1935
  R7 := 0;                                                              1936
  FOR R2 := 1 STEP 1 UNTIL NUMTERMINALS DO                              1937
  BEGIN                                                                 1938
    SYM := R2; ISVALIDLA; R7 := R7 SHLL 1;                              1939
    IF ISVALID THEN R7 := R7 OR #1;                                     1940
    R6 := R6 + 1;                                                       1941
    IF R2 = NUMTERMINALS AND R1 = TEMP THEN SET(ARRAYEND);              1942
    IF R6= 8 THEN                                                       1943
    BEGIN                                                               1944
      NUMLINE := R7; BUILDHEXLINE; PUNCHCARD;                           1945
      R6 := BYTECNT + 1; BYTECNT := R6; R6 := 0; R7 := 0;               1946
    END;                                                                1947
  END;                                                                  1948
  IF R6 > 0 THEN                                                        1949
  BEGIN                                                                 1950
    R2 := NUMTERMINALS AND #7; R3 := 7 - R2;                            1951
    R7 := R7 SHLL R3; R2 := BYTECNT + 1; BYTECNT := R2;                 1952
    NUMLINE := R7; BUILDHEXLINE; PUNCHCARD;                             1953
  END;                                                                  1954
  WRITE; PUNCH; MVC(80,WBUF,BLANK); WRITE; PUNCH;                       1955
END;                                                                    1956
WRITE; PUNCH;                                                           1957
```

```
   R14 := SAVE14;                                                          1958
 END;  COMMENT  END OF PUNCH LASETS;                                       1959
                                                                           1960
                                                                           1961
SAVE14 := R14; RESET(LL);                                                  1962
MVC(21,WBUF(2),"INTEGER NUMTERMINALS ="); R1 := NUMTERMINALS;              1963
CVD(R1,CONWORK); UNPK(3,7,WBUF(25),CONWORK); SETZONE(WBUF(28));            1964
MVC(0,WBUF(29),";"); R0 := @WBUF; WRITE; PUNCH;                            1965
MVC(29,WBUF,BLANK); MVC(15,WBUF(2),"INTEGER NUMNTS =");                    1966
R1 := NUMNTS; CVD(R1,CONWORK); UNPK(3,7,WBUF(19),CONWORK);                 1967
SETZONE(WBUF(22)); MVC(0,WBUF(23),";"); WRITE; PUNCH;                      1968
MVC(23,WBUF,BLANK); MVC(16,WBUF(2),"INTEGER NUMSYMS =");                   1969
R1 := NUMSYMS + 1; CVD(R1,CONWORK); UNPK(3,7,WBUF(20),CONWORK);            1970
SETZONE(WBUF(23));                                                         1971
MVC(0,WBUF(24),";"); WRITE; PUNCH; MVC(24,WBUF,BLANK);                     1972
WRITE; WRITE; PUNCH; PUNCH;                                                1973
MVC(30,WBUF(2),"GLOBAL DATA SEGTABLE BASE R12; ");                         1974
WRITE; PUNCH; MVC(30,WBUF(2),BLANK);                                       1975
WRITE; PUNCH;                                                              1976
R12 := BASE005;                                                            1977
R11 := BASE004;                                                            1978
MVC(8,V(0),"ERRORSYM ");                                                   1979
R1 := VLENGTH; ARRAYLENGTH := R1; R1 := @V;                                1980
ADDRESS := R1; R1 := 8; NUMBITS := R1; MVC(6,TBUF,"VSTRING");              1981
R1 := 6; DECLENGTH := R1; PUNCHARRAY;                                      1982
R1 := NUMTERMINALS + NUMNTS; ARRAYLENGTH := R1;                            1983
R1 := @LOCLENGTH; ADDRESS := R1; R1 := 32; NUMBITS := R1;                  1984
SET(LL);                                                                   1985
MVC(8,TBUF,"LOCLENGTH"); R1 := 8; DECLENGTH := R1; PUNCHARRAY;             1986
RESET(LL); R11 := BASE006;                                                 1987
MVC(20,WBUF(2),"COMMENT  THE DPDA HAS");                                   1988
MVC(11,WBUF(29),"READ STATES;");                                           1989
R1 := NUMREADSTATES + 1; R2 := 24;                                         1990
PUNCHCOMMENT;                                                              1991
R1 := @TREADSTART; ADDRESS := R1; R1 := 16; NUMBITS := R1;                 1992
R1 := NUMREADSTATES; ARRAYLENGTH := R1; MVC(8,TBUF,"READSTART");           1993
R5 := 8; DECLENGTH := R5; PUNCHARRAY;                                      1994
R1 := @TRDNUM; ADDRESS := R1; R1 := 8; NUMBITS := R1;                      1995
MVC(4,TBUF,"RDNUM"); R1 := 4; DECLENGTH := R1; PUNCHARRAY;                 1996
R1 := @TSYMLIST; ADDRESS := R1; R1 := NUMREADSTATES;                       1997
R2 := R2 - R2; IC(R2,TRDNUM(R1)); R1 := R1 SHLL 1;                         1998
R2 := R2 + TREADSTART(R1); ARRAYLENGTH := R2;                              1999
MVC(6,TBUF,"SYMLIST"); R1 := 6; DECLENGTH := R1; PUNCHARRAY;               2000
R12 := BASEREG008; R1 := @TSTATELIST; ADDRESS:= R1;                        2001
R1 := 16; NUMBITS := R1; MVC(8,TBUF,"STATELIST");                          2002
R1 := 8; DECLENGTH := R1; PUNCHARRAY;                                      2003
                                                                           2004
MVC(20,WBUF(2),"COMMENT  THE DPDA HAS");                                   2005
```

```
MVC(13,WBUF(29),"REDUCE STATES;"); R1 := NUMPRODS; R2 := 24;
PUNCHCOMMENT;
R12 := BASE005;
FOR R1 := 1 STEP 1 UNTIL NUMPRODS DO
BEGIN
  R2 := R2 - R2; IC(R2,RTPTSIZE(R1));
  IF R2 > 0 THEN
  BEGIN
    R2 := R2 - 1; STC(R2,RTPTSIZE(R1));
  END;
END;
R1 := @RTPTSIZE; ADDRESS := R1; R1 := NUMPRODS;
ARRAYLENGTH := R1; R1 := 8; NUMBITS := R1;
MVC(7,TBUF,"NUMTOPOP"); R1 := 7; DECLENGTH := R1; PUNCHARRAY;
R1 := @REDUCESUCC; ADDRESS := R1; R1 := 16; NUMBITS := R1;
MVC(9,TBUF,"REDUCESUCC"); R1 := 9; DECLENGTH := R1; PUNCHARRAY;

MVC(20,WBUF(2),"COMMENT  THE DPDA HAS");
MVC(17,WBUF(29),"LOOK AHEAD STATES;");
R1 := NUMLASTATES; R2 := 24; PUNCHCOMMENT;

R1 := @LASYMNUM; ADDRESS := R1; R1 := 8; NUMBITS := R1;
R1 := NUMLASTATES - 1; ARRAYLENGTH := R1; R1 := 7;
DECLENGTH := R1; MVC(7,TBUF,"LASYMNUM"); PUNCHARRAY;
R1 := @SUCCSTATE; ADDRESS := R1; R1 := 16; NUMBITS := R1;
MVC(8,TBUF,"SUCCSTATE"); R1 := 8; DECLENGTH := R1; PUNCHARRAY;
R1 := @FAILSTATE; ADDRESS := R1;
MVC(8,TBUF,"FAILSTATE"); PUNCHARRAY;

PUNCHLASETS;

MVC(20,WBUF(2),"COMMENT  THE DPDA HAS");
MVC(15,WBUF(29),"LOOKBACK STATES;");
R1 := NUMLBSTATES; R2 := 24; PUNCHCOMMENT;

R1 := NUMLBSTATES - 1;
IF R1 > 0 THEN
BEGIN
  ARRAYLENGTH := R1; R1 := @LBSTART; ADDRESS := R1; R1 := 8;
  NUMBITS := R1; MVC(6,TBUF,"LBSTART"); R1 := 6;
  DECLENGTH := R1; PUNCHARRAY;
  R1 := @LBNUM; ADDRESS := R1; MVC(4,TBUF,"LBNUM");
  R1 := 4; DECLENGTH := R1;
  PUNCHARRAY;
END ELSE
BEGIN
  R4 := 0; R3 := SEGPT SHLL 2; SEGTABLE(R3) := R4;
  R3 := R3 + 4; SEGTABLE(R3) := R4;
```

```
      R3 := R3 SHRL 2 + 1; SEGPT := R3;                                    2054
    END;                                                                   2055
    R1 := NUMLBSTATES - 1; R2 := R2 - R2; IC(R2,LBNUM(R1));                2056
    R3 := R3 - R3; IC(R3,LBSTART(R1)); R2 := R2 + R3;                      2057
    ARRAYLENGTH := R2;                                                     2058
    R12 := BASEREG008; R1 := @LBSTATE; ADDRESS := R1; R1 := 16;            2059
    NUMBITS := R1; MVC(6,TBUF,"LBSTATE"); R1 := 6;                         2060
    DECLENGTH := R1; PUNCHARRAY;                                           2061
    R1 := @RESUMESTATE; ADDRESS:= R1;                                      2062
    MVC(10,TBUF,"RESUMESTATE"); R1 := 10; DECLENGTH := R1;                 2063
    PUNCHARRAY;                                                            2064
    MVC(41,WBUF(2),"COMMENT  THE SYMBOLS ACCESSING THE STATES;");          2065
    WRITE; PUNCH; MVC(43,WBUF,BLANK); WRITE; PUNCH;                        2066
    R8 := BASE010; R1 := @SYMBEFORE; ADDRESS := R1;                        2067
    R1 := NUMREADSTATES; ARRAYLENGTH := R1; R1 := 8; NUMBITS := R1;        2068
    MVC(12,TBUF,"SYMBEFOREREAD"); R1 := 12; DECLENGTH := R1;               2069
    PUNCHARRAY;                                                            2070
    R1 := ADDRESS + #300; ADDRESS := R1; R1 := NUMLASTATES - 1;            2071
    ARRAYLENGTH := R1; MVC(10,TBUF,"SYMBEFORELA"); R1 := 10;               2072
    DECLENGTH := R1; PUNCHARRAY;                                           2073
    MVC(10,WBUF(2),"CLOSE BASE;"); WRITE; PUNCH;                           2074
    MVC(10,WBUF(2),BLANK); WRITE; PUNCH;                                   2075
    MVC(28,WBUF(2),"ARRAY 18 INTEGER SEGTABLE = (");                       2076
    WRITE; PUNCH; MVC(28,WBUF(2),BLANK); R2 := 17 SHLL 2;                  2077
    R1 := 32; NUMBITS := R1;                                               2078
    FOR R1 := 0 STEP 4 UNTIL R2 DO                                         2079
    BEGIN                                                                  2080
      R3 := SEGTABLE(R1); NUMLINE := R3;                                   2081
      BUILDHEXLINE; MVC(8,WBUF(10),HEXLINE);                               2082
      IF R1 = R2 THEN MVC(1,WBUF(19),");")                                 2083
      ELSE MVC(0,WBUF(19),",");                                            2084
      WRITE; PUNCH; MVC(10,WBUF(10),BLANK);                                2085
    END;                                                                   2086
    MVC(42,WBUF(2),"COMMENT  END OF CARDS PUNCHED BY THE SLR(1)");         2087
    MVC(15,WBUF(46),"SYNTAX ANALYZER;"); WRITE; PUNCH;                     2088
    MVC(61,WBUF,BLANK); WRITE; PUNCH;                                      2089
    R14 := SAVE14;                                                         2090
  END;  COMMENT  END OF PUNCHDPDA;                                         2091
                                                                           2092
COMMENT  MAIN PROGRAM;                                                     2093
                                                                           2094
COMMENT  THE FOLLOWING 5 LINES FIND AND SET BASE ADDRESS FOR DATA          2095
SEGMENTS.  THEY SHOULD BE CHANGED ONLY WHEN THE DATA SEGMENTS              2096
HAVE BEEN CHANGED;                                                         2097
                                                                           2098
                                                                           2099
R1 := @TSTATELIST; BASEREG008 := R1;                                       2100
R1 := R11 - #D00; BASE005 := R1;                                           2101
```

```
R1 := R1 - #1000; BASE004 := R1; BASE006 := R11;                          2102
R12 := BASE005;                                                           2103
R1 := R8 - #E00; BASE010 := R1; BASE011 := R8; R8 := BASE010;             2104
STM(R7,R11,SAVEBASEREGS);                                                 2105
SET(MOREGRAMMERS);                                                        2106
WHILE MOREGRAMMERS DO                                                     2107
BEGIN                                                                     2108
  R1 := 0; ERRCOUNT := R1;                                                2109
  RESET(INPUTLIST);  COMMENT  DO NOT LIST GRAMMER CARDS;                  2110
  SET(GRAMMERLIST);  COMMENT  LIST REFORMATTED GRAMMER;                   2111
  RESET(CONFIGLIST);  COMMENT  DO NOT LIST CONFIGURATION SETS;            2112
  SET(FSMLIST);  COMMENT  LIST CHARACTERISTIC FSM;                        2113
  SET(LASETSLIST);  COMMENT  LIST LOOK-AHEAD SETS;                        2114
  SET(DPDALIST);  COMMENT  LIST THE DPDA;                                 2115
  RESET(PUNCHDECK);  COMMENT  DO NOT PUNCH THE DPDA;                      2116
                                                                          2117
  MVC(24,WBUF,"START READING THE GRAMMAR");                               2118
  R0 := @WBUF; WRITE; MVC(24,WBUF,BLANK);                                 2119
  READG;                                                                  2120
  MVC(24,WBUF,"THE GRAMMAR HAS BEEN READ");                               2121
  R0 := @WBUF; WRITE; MVC(24,WBUF,BLANK);                                 2122
  R1 := ERRCOUNT;                                                         2123
  IF R1¬=0 THEN                                                           2124
  BEGIN                                                                   2125
    IF R1 = 1 THEN  MVC(19,WBUF,"THERE WAS ONE ERROR,")                   2126
    ELSE                                                                  2127
    BEGIN                                                                 2128
      MVC(19,WBUF,"THERE WERE    ERRORS,");                               2129
      CVD(R1,CONWORK); UNPK(1,7,WBUF(11),CONWORK);                        2130
      SETZONE(WBUF(12));                                                  2131
    END;                                                                  2132
    WRITE;                                                                2133
    MVC(42,WBUF,"THEREFORE THE GRAMMAR WILL NOT BE ANALYZED.");           2134
    WRITE;                                                                2135
    MVC(41,WBUF,"THE GRAMMAR IS LISTED ABOVE FOR DEBUGGING");             2136
    MVC(8,WBUF(43),"PURPOSES.");                                          2137
    WRITE; MVC(50,WBUF,BLANK); WRITE; WRITE;                              2138
  END;                                                                    2139
  R1 := ERRCOUNT;                                                         2140
  IF R1 = 0 THEN                                                          2141
  BEGIN                                                                   2142
    MVC(31,WBUF,"START CONSTRUCTING THE GRAMMAR'S");                      2143
    MVC(18,WBUF(35),"CHARACTERISTIC FSM.");                               2144
    R0 := @WBUF; WRITE; MVC(53,WBUF,BLANK);                               2145
    SET(ISLR0); SET(ISLR1);                                               2146
    LM(R7,R11,SAVEBASEREGS);                                              2147
    COMPUTECFSM;                                                          2148
    MVC(42,WBUF,"THE CFSM FOR THE GRAMMARS HAS BEEN COMPUTED.");          2149
```

```
      R0 := @WBUF; WRITE; MVC(42,WBUF,BLANK);                                  2150
      PAGE;                                                                    2151
      R1 := ERRCOUNT;                                                          2152
      IF FSMLIST OR R1 ¬= 0 THEN PRINTCFSM;                                    2153
    END;                                                                       2154
    R1 := ERRCOUNT;                                                            2155
    IF R1 ¬= 0 THEN                                                            2156
    BEGIN                                                                      2157
      MVC(27,WBUF,"SINCE THE CFSM IS IN ERROR,");                              2158
      MVC(27,WBUF(28),"ANALYSIS WILL BE TERMINATED.");                         2159
      WRITE;  MVC(54,WBUF,BLANK);                                              2160
    END ELSE                                                                   2161
    BEGIN                                                                      2162
      IF ISLR0 THEN                                                            2163
      BEGIN                                                                    2164
        MVC(32,WBUF,"SURPRISE    THE GRAMMAR IS LR(0).");                      2165
        WRITE; MVC(32,WBUF,BLANK);                                             2166
      END ELSE                                                                 2167
      BEGIN                                                                    2168
        MVC(24,WBUF,"THE GRAMMAR IS NOT LR(0).");                              2169
        MVC(24,WBUF(27),"LOOK-AHEAD MUST BE ADDED.");                          2170
        WRITE; MVC(60,WBUF,BLANK); WRITE; WRITE;                               2171
        MVC(31,WBUF,"START COMPUTING LOOK-AHEAD SETS.");                       2172
        WRITE; MVC(31,WBUF,BLANK);                                             2173
        LOOKAHEADANDDPDA;                                                      2174
      END;                                                                     2175
    END;                                                                       2176
    R1 := ERRCOUNT;                                                            2177
    IF R1 ¬= 0 THEN                                                            2178
    BEGIN                                                                      2179
      MVC(38,WBUF,"SINCE THE LOOK-AHEAD SETS ARE IN ERROR,");                  2180
      MVC(28,WBUF(40),"EXECUTION WILL BE TERMINATED.");                        2181
      WRITE; MVC(58,WBUF,BLANK);                                               2182
    END ELSE                                                                   2183
    BEGIN                                                                      2184
      MVC(24,WBUF,"START COMPUTING THE DPDA.");                                2185
      WRITE; MVC(24,WBUF,BLANK);                                               2186
      COMPUTEDPDA;                                                             2187
    END;                                                                       2188
    PAGE;                                                                      2189
    IF ¬PUNCHDECK THEN                                                         2190
    BEGIN                                                                      2191
      MVC(22,WBUF,"NO DPDA DECK REQUESTED.");                                  2192
      WRITE; MVC(22,WBUF,BLANK);                                               2193
    END ELSE PUNCHDPDA;                                                        2194
    WRITE; WRITE; MVC(8,WBUF,"ALL DONE."); WRITE;                              2195
  END;                                                                         2196
BAILOUT:                                                                       2197
```

END.

APPENDIX B

SAMPLE GRAMMAR FOR THE SYNTAX ANALYZER

```
@P
@C
@I
<PROGRAM> <STATEMENT LIST> END
<STATEMENT LIST> <STATEMENT>
     <STATEMENT LIST> <STATEMENT>
<STATEMENT> <ASSIGNMENT> ;
<ASSIGNMENT> <VARIABLE> = <EXPR>
<EXPR> <ARITH EXPR>
<ARITH EXPR> <TERM>
     <ARITH EXPR> + <TERM>
     <ARITH EXPR> - <TERM>
<TERM> <PRIMARY>
     <TERM> * <PRIMARY>
     <TERM> / <PRIMARY>
<PRIMARY> <VARIABLE>
     <NUMBER>
<VARIABLE> <IDENTIFIER>
```

APPENDIX C

SYNTAX ANALYZER OUTPUT

```
START READING THE GRAMMAR
                    <PROGRAM> <STATEMENT LIST> END
                    <STATEMENT LIST> <STATEMENT>
                         <STATEMENT LIST> <STATEMENT>
                    <STATEMENT> <ASSIGNMENT> ;
                    <ASSIGNMENT> <VARIABLE> = <EXPR>
                    <EXPR> <ARITH EXPR>
                    <ARITH EXPR> <TERM>
                         <ARITH EXPR> + <TERM>
                         <ARITH EXPR> - <TERM>
                    <TERM> <PRIMARY>
                         <TERM> * <PRIMARY>
                         <TERM> / <PRIMARY>
                    <PRIMARY> <VARIABLE>
                         <NUMBER>
                    <VARIABLE> <IDENTIFIER>
```

THE VOCABULARY

	TERMINAL SYMBOLS	NONTERMINALS
0001	_|_	<PROGRAM>
0002	END	<STATEMENT LIST>
0003	;	<STATEMENT>
0004	=	<ASSIGNMENT>
0005	+	<VARIABLE>
0006	-	<EXPR>
0007	*	<ARITH EXPR>
0008	/	<TERM>
0009	<NUMBER>	<PRIMARY>
0010	<IDENTIFIER>	

THE GOAL SYMBOL IS <PROGRAM>

THE PRODUCTIONS

```
0001  <PROGRAM>   ::=  <STATEMENT LIST>  END

0002  <STATEMENT LIST>   ::=  <STATEMENT>
0003                         |  <STATEMENT LIST>  <STATEMENT>

0004  <STATEMENT>   ::=  <ASSIGNMENT>  ;

0005  <ASSIGNMENT>   ::=  <VARIABLE>  =  <EXPR>

0006  <EXPR>   ::=  <ARITH EXPR>

0007  <ARITH EXPR>   ::=  <TERM>
0008                       |  <ARITH EXPR>  +  <TERM>
0009                       |  <ARITH EXPR>  -  <TERM>

0010  <TERM>   ::=  <PRIMARY>
0011               |  <TERM>  *  <PRIMARY>
0012               |  <TERM>  /  <PRIMARY>

0013  <PRIMARY>   ::=  <VARIABLE>
0014                   |  <NUMBER>

0015  <VARIABLE>   ::=  <IDENTIFIER>
```

```
SCME STATISTICS ON THE GRAMMAR:

     NUMBER OF TERMINAL SYMBOLS = 0010
     NUMBER OF NONTERMINAL SYMBOLS = 0009
          TOTAL NUMBER OF SYMBOLS = 0019

     NUMBER OF PRODUCTIONS = 0015
THE GRAMMAR HAS BEEN READ
```

```
START CONSTRUCTING THE GRAMMAR'S   CHARACTERISTIC FSM.
THE CONFIGURATION SET FOR STATE    0000 IS:
    0001  <SYSTEMGS>  -> . _|_  <PROGRAM>  _|_

THE CONFIGURATION SET FOR STATE    0001 IS:
    0001  <SYSTEMGS>  ->  _|_ . <PROGRAM>  _|_
    0002  <PROGRAM>  -> . <STATEMENT LIST>  END
    0003  <STATEMENT LIST>  -> . <STATEMENT LIST>  <STATEMENT>
    0004  <STATEMENT LIST>  -> . <STATEMENT>
    0005  <STATEMENT>  -> . <ASSIGNMENT>  ;
    0006  <ASSIGNMENT>  -> . <VARIABLE>  =  <EXPR>
    0007  <VARIABLE>  -> . <IDENTIFIER>


THE CONFIGURATION SET FOR STATE    0002 IS:
    0001  <SYSTEMGS>  ->  _|_  <PROGRAM> . _|_


THE CONFIGURATION SET FOR STATE    0003 IS:
    0001  <PROGRAM>  ->  <STATEMENT LIST> . END
    0002  <STATEMENT LIST>  ->  <STATEMENT LIST> . <STATEMENT>
    0003  <STATEMENT>  -> . <ASSIGNMENT>  ;
    0004  <ASSIGNMENT>  -> . <VARIABLE>  =  <EXPR>
    0005  <VARIABLE>  -> . <IDENTIFIER>


THE CONFIGURATION SET FOR STATE    0004 IS:
    0001  <STATEMENT>  ->  <ASSIGNMENT> . ;


THE CONFIGURATION SET FOR STATE    0005 IS:
    0001  <ASSIGNMENT>  ->  <VARIABLE> . =  <EXPR>


THE CONFIGURATION SET FOR STATE    0006 IS:
    0001  <ASSIGNMENT>  ->  <VARIABLE>  = . <EXPR>
    0002  <EXPR>  -> . <ARITH EXPR>
    0003  <ARITH EXPR>  -> . <ARITH EXPR>  +  <TERM>
    0004  <ARITH EXPR>  -> . <ARITH EXPR>  -  <TERM>
    0005  <ARITH EXPR>  -> . <TERM>
    0006  <TERM>  -> . <TERM>  *  <PRIMARY>
    0007  <TERM>  -> . <TERM>  /  <PRIMARY>
    0008  <TERM>  -> . <PRIMARY>
    0009  <PRIMARY>  -> . <VARIABLE>
    0010  <PRIMARY>  -> . <NUMBER>
    0011  <VARIABLE>  -> . <IDENTIFIER>
```

```
THE CONFIGURATION SET FOR STATE    0007 IS:
    0001  <EXPR>  ->  <ARITH EXPR> .
    0002  <ARITH EXPR>  ->  <ARITH EXPR> . +  <TERM>
    0003  <ARITH EXPR>  ->  <ARITH EXPR> . -  <TERM>

THE CONFIGURATION SET FOR STATE    0008 IS:
    0001  <ARITH EXPR>  ->  <TERM> .
    0002  <TERM>  ->  <TERM> . *  <PRIMARY>
    0003  <TERM>  ->  <TERM> . /  <PRIMARY>

THE CONFIGURATION SET FOR STATE    0009 IS:
    0001  <ARITH EXPR>  ->  <ARITH EXPR>  + . <TERM>
    0002  <TERM>  -> . <TERM>  *  <PRIMARY>
    0003  <TERM>  -> . <TERM>  /  <PRIMARY>
    0004  <TERM>  -> . <PRIMARY>
    0005  <PRIMARY>  -> . <VARIABLE>
    0006  <PRIMARY>  -> . <NUMBER>
    0007  <VARIABLE>  -> . <IDENTIFIER>

THE CONFIGURATION SET FOR STATE    0010 IS:
    0001  <ARITH EXPR>  ->  <ARITH EXPR>  - . <TERM>
    0002  <TERM>  -> . <TERM>  *  <PRIMARY>
    0003  <TERM>  -> . <TERM>  /  <PRIMARY>
    0004  <TERM>  -> . <PRIMARY>
    0005  <PRIMARY>  -> . <VARIABLE>
    0006  <PRIMARY>  -> . <NUMBER>
    0007  <VARIABLE>  -> . <IDENTIFIER>

THE CONFIGURATION SET FOR STATE    0011 IS:
    0001  <TERM>  ->  <TERM>  * . <PRIMARY>
    0002  <PRIMARY>  -> . <VARIABLE>
    0003  <PRIMARY>  -> . <NUMBER>
    0004  <VARIABLE>  -> . <IDENTIFIER>

THE CONFIGURATION SET FOR STATE    0012 IS:
    0001  <TERM>  ->  <TERM>  / . <PRIMARY>
    0002  <PRIMARY>  -> . <VARIABLE>
    0003  <PRIMARY>  -> . <NUMBER>
    0004  <VARIABLE>  -> . <IDENTIFIER>

THE CONFIGURATION SET FOR STATE    0013 IS:
    0001  <ARITH EXPR>  ->  <ARITH EXPR>  +  <TERM> .
```

```
    0002  <TERM>  ->  <TERM> . *  <PRIMARY>
    0003  <TERM>  ->  <TERM> . /  <PRIMARY>

THE CONFIGURATION SET FOR STATE    0014 IS:
    0001  <ARITH EXPR>  ->  <ARITH EXPR>  -  <TERM> .
    0002  <TERM>  ->  <TERM> . *  <PRIMARY>
    0003  <TERM>  ->  <TERM> . /  <PRIMARY>
```

```
THE CFSM FOR THE GRAMMARS HAS BEEN COMPUTED
THE CFSM FOR THE GRAMMAR IS AS FOLLOWS

STATE READ LINEAR       0000:    ACCESSING SYMBOL IS  <SYSTEMGS>)

                                 _|_  -> READ LINEAR         0001
          ----------------------------------------------------

STATE READ LINEAR       0001:    ACCESSING SYMBOL IS  _|_)

                        <IDENTIFIER>  -> REDUCE              0015
          ----------------------------------------------------
                           <PROGRAM>  -> READ LINEAR         0002
                    <STATEMENT LIST>  -> READ LINEAR         0003
                         <STATEMENT>  -> REDUCE              0002
                        <ASSIGNMENT>  -> READ LINEAR         0004
                          <VARIABLE>  -> READ LINEAR         0005

STATE READ LINEAR       0002:    ACCESSING SYMBOL IS  <PROGRAM>)

                                 _|_  -> REDUCE              0016
          ----------------------------------------------------

STATE READ LINEAR       0003:    ACCESSING SYMBOL IS  <STATEMENT LIST>)

                                 END  -> REDUCE              0001
                        <IDENTIFIER>  -> REDUCE              0015
          ----------------------------------------------------
                         <STATEMENT>  -> REDUCE              0003
                        <ASSIGNMENT>  -> READ LINEAR         0004
                          <VARIABLE>  -> READ LINEAR         0005

STATE READ LINEAR       0004:    ACCESSING SYMBOL IS  <ASSIGNMENT>)

                                   ;  -> REDUCE              0004
          ----------------------------------------------------

STATE READ LINEAR       0005:    ACCESSING SYMBOL IS  <VARIABLE>)

                                   =  -> READ LINEAR         0006
          ----------------------------------------------------
```

```
STATE READ LINEAR       0006:    ACCESSING SYMBOL IS  =)

                        <NUMBER>  -> REDUCE              0014
                    <IDENTIFIER>  -> REDUCE              0015
                ------------------------------------------
                      <VARIABLE>  -> REDUCE              0013
                          <EXPR>  -> REDUCE              0005
                    <ARITH EXPR>  -> LOOK AHEAD ORD      0000
                          <TERM>  -> LOOK AHEAD ORD      0001
                       <PRIMARY>  -> REDUCE              0010

READ STATE   0007 IS INADEQUATE   THE FOLLOWING LOOK-AHEAD IS NECESSARY.

LOOK AHEAD ORD    0000  :       LA SYM NUM = 0001)
LA SET OF                      <EXPR>  -> REDUCE        0006
                              DEFAULT  -> READ LINEAR   0007 )

STATE READ LINEAR       0007:)   ACCESSING SYMBOL IS  <ARITH EXPR>)

                                 +  -> READ LINEAR       0009
                                 -  -> READ LINEAR       0010
                ------------------------------------------

READ STATE   0008 IS INADEQUATE   THE FOLLOWING LOOK-AHEAD IS NECESSARY.

LOOK AHEAD ORD    0001  :       LA SYM NUM = 0001)
LA SET OF                   <ARITH EXPR>  -> REDUCE      0007 )
                              DEFAULT  -> READ LINEAR   0008

STATE READ LINEAR       0008:    ACCESSING SYMBOL IS  <TERM>)

                                 *  -> READ LINEAR       0011
                                 /  -> READ LINEAR       0012
                ------------------------------------------

STATE READ LINEAR       0009:    ACCESSING SYMBOL IS  +)

                        <NUMBER>  -> REDUCE              0014
                    <IDENTIFIER>  -> REDUCE              0015
```

```
                    ----------------------------------------------
                                <VARIABLE>  -> REDUCE              0013
                                   <TERM>   -> LOOK AHEAD ORD      0002
                                 <PRIMARY>  -> REDUCE              0010

STATE READ LINEAR       0010:     ACCESSING SYMBOL IS  -)

                                   <NUMBER>  -> REDUCE             0014
                                <IDENTIFIER> -> REDUCE             0015
                    ----------------------------------------------
                                <VARIABLE>  -> REDUCE              0013
                                   <TERM>   -> LOOK AHEAD ORD      0003
                                 <PRIMARY>  -> REDUCE              0010

STATE READ LINEAR       0011:     ACCESSING SYMBOL IS  *)

                                   <NUMBER>  -> REDUCE             0014
                                <IDENTIFIER> -> REDUCE             0015
                    ----------------------------------------------
                                <VARIABLE>  -> REDUCE              0013
                                 <PRIMARY>  -> REDUCE              0011

STATE READ LINEAR       0012:     ACCESSING SYMBOL IS  /)

                                   <NUMBER>  -> REDUCE             0014
                                <IDENTIFIER> -> REDUCE             0015
                    ----------------------------------------------
                                <VARIABLE>  -> REDUCE              0013
                                 <PRIMARY>  -> REDUCE              0012

READ STATE   0013 IS INADEQUATE   THE FOLLOWING LOOK-AHEAD IS NECESSARY.

LOOK AHEAD ORD    0002  :       LA SYM NUM = 0001)
LA SET OF                   <ARITH EXPR>  -> REDUCE              0008
                                 DEFAULT  -> READ LINEAR         0013

STATE READ LINEAR       0013:     ACCESSING SYMBOL IS  <TERM>)

                                      *  -> READ LINEAR          0011
                                      /  -> READ LINEAR          0012
                    ----------------------------------------------
```

```
READ STATE   0014 IS INADEQUATE   THE FOLLOWING LOOK-AHEAD IS NECESSARY.

LOOK AHEAD ORD    0003  :       LA SYM NUM = 0001)
LA SET OF                <ARITH EXPR>  -> REDUCE          0009
                               DEFAULT  -> READ LINEAR     0014

STATE READ LINEAR       0014:    ACCESSING SYMBOL IS  <TERM>)

                                    *  -> READ LINEAR     0011
                                    /  -> READ LINEAR     0012
```

```
THE GRAMMAR IS NOT LR(0).  LOOK-AHEAD MUST BE ADDED.

START COMPUTING LOOK-AHEAD SETS.
LOOK-AHEAD SETS HAVE BEEN COMPUTED.
THE GRAMMAR IS LR(1).
```

T H E L O O K - A H E A D S E T S

NONTERMINAL SYMBOL	TERMINAL SYMBOLS 1	2	3	4	5	6	7	8	9	10
<PROGRAM>	1									
<STATEMENT L		1								1
<STATEMENT>		1								1
<ASSIGNMENT>			1							
<VARIABLE>			1	1	1	1	1	1		
<EXPR>			1							
<ARITH EXPR>			1		1	1				
<TERM>			1		1	1	1	1		
<PRIMARY>			1		1	1	1	1		

START COMPUTING THE DPDA.
THE DPDA HAS BEEN COMPUTED,

THE DPDA FOR THE GRAMMAR CONSISTS OF THE TERMINAL TRANSITIONS OF THE CFSM PLUS THE FOLLOWING REDUCE AND LOOK-BACK TRANSITIONS

```
STATE REDUCE 1:      POP 01          ->  READ LINEAR       0002
STATE REDUCE 2:      POP 00          ->  READ LINEAR       0003
STATE REDUCE 3:      POP 01          ->  READ LINEAR       0003
STATE REDUCE 4:      POP 01          ->  LOOK BACK         0000
STATE REDUCE 5:      POP 02          ->  READ LINEAR       0004
STATE REDUCE 6:      POP 00          ->  REDUCE            0005
STATE REDUCE 7:      POP 00          ->  LOOK AHEAD ORD    0000
STATE REDUCE 8:      POP 02          ->  LOOK AHEAD ORD    0000
STATE REDUCE 9:      POP 02          ->  LOOK AHEAD ORD    0000
STATE REDUCE 10:     POP 00          ->  LOOK BACK         0002
STATE REDUCE 11:     POP 02          ->  LOOK BACK         0002
STATE REDUCE 12:     POP 02          ->  LOOK BACK         0002
STATE REDUCE 13:     POP 00          ->  LOOK BACK         0003
STATE REDUCE 14:     POP 00          ->  LOOK BACK         0003
STATE REDUCE 15:     POP 00          ->  LOOK BACK         0001
STATE REDUCE 16:     POP 02          ->  EXIT              0000
STATE LOOK BACK 00:
                READ LINEAR      0001->  REDUCE            0002
                DEFAULT             ->  REDUCE            0003

STATE LOOK BACK 01:
                READ LINEAR      0001->  READ LINEAR       0005
                READ LINEAR      0003->  READ LINEAR       0005
                DEFAULT             ->  REDUCE            0013
```

```
STATE LOCK BACK 02:
                READ LINEAR         0006->  LOOK AHEAD ORD      0001
                READ LINEAR         0009->  LOOK AHEAD ORD      0002
                DEFAULT                ->   LOOK AHEAD ORD      0003

STATE LOCK BACK 03:
                READ LINEAR         0011->  REDUCE              0011
                READ LINEAR         0012->  REDUCE              0012
                DEFAULT                ->   REDUCE              0010
```

```
INTEGER NUMTERMINALS = 0010;
INTEGER NUMNTS = 0009;
INTEGER NUMSYMS = 0020;

GLOBAL DATA SEGTABLE BASE R12;

ARRAY 0133 BYTE VSTRING = (#C5X, #D9X, #D9X, #D6X, #D9X, #E2X, #E8X,
       #D4X, #40X, #6EX, #6DX, #4FX, #6DX, #4CX, #D7X, #D9X, #D6X,
       #C7X, #D9X, #C1X, #D4X, #6EX, #4CX, #E2X, #E3X, #C1X, #E3X,
       #C5X, #D4X, #C5X, #D5X, #E3X, #40X, #D3X, #C9X, #E2X, #E3X,
       #6EX, #C5X, #D5X, #C4X, #4CX, #E2X, #E3X, #C1X, #E3X, #C5X,
       #D4X, #C5X, #D5X, #E3X, #6EX, #4CX, #C1X, #E2X, #E2X, #C9X,
       #C7X, #D5X, #D4X, #C5X, #D5X, #E3X, #6EX, #5EX, #4CX, #E5X,
       #C1X, #D9X, #C9X, #C1X, #C2X, #D3X, #C5X, #6EX, #7EX, #4CX,
       #C5X, #E7X, #D7X, #D9X, #6EX, #4CX, #C1X, #D9X, #C9X, #E3X,
       #C8X, #40X, #C5X, #E7X, #D7X, #D9X, #6EX, #4CX, #E3X, #C5X,
       #D9X, #D4X, #6EX, #4EX, #60X, #4CX, #D7X, #D9X, #C9X, #D4X,
       #C1X, #D9X, #E8X, #6EX, #5CX, #61X, #4CX, #D5X, #E4X, #D4X,
       #C2X, #C5X, #D9X, #6EX, #4CX, #C9X, #C4X, #C5X, #D5X, #E3X,
       #C9X, #C6X, #C9X, #C5X, #D9X, #6EX);

ARRAY 0020 INTEGER LOCLENGTH = (#0000000A, #00000283, #00000983,
       #00001001, #000012C1, #00001901, #00001941, #00001BC1,
       #00001C01, #00001C48, #00001E4C, #00000349, #00000590,
       #00000A4B, #00000D0C, #0000104A, #00001306, #0000148C,
       #00001786, #00001989);

COMMENT  THE DPDA HAS 0015 READ STATES;

ARRAY 0015 SHORT INTEGER READSTART = (#0000S, #0002S, #0004S, #0006S,
       #0009S, #000BS, #000DS, #0010S, #0013S, #0016S, #0019S,
       #001CS, #001FS, #0022S, #0025S);

ARRAY 0015 BYTE RDNUM = (#01X, #01X, #01X, #02X, #01X, #01X, #02X,
       #02X, #02X, #02X, #02X, #02X, #02X, #02X, #02X);

ARRAY 0040 BYTE SYMLIST = (#01X, #00X, #0AX, #00X, #01X, #00X, #02X,
       #0AX, #00X, #03X, #00X, #04X, #00X, #09X, #0AX, #00X, #05X,
       #06X, #00X, #07X, #08X, #00X, #09X, #0AX, #00X, #09X, #0AX,
       #00X, #09X, #0AX, #00X, #09X, #0AX, #00X, #07X, #08X, #00X,
       #07X, #08X, #00X);

ARRAY 0040 SHORT INTEGER STATELIST = (#0001S, #06FFS, #020FS, #06FFS,
       #0210S, #06FFS, #0201S, #020FS, #06FFS, #0204S, #06FFS,
       #0006S, #06FFS, #020ES, #020FS, #06FFS, #0009S, #000AS,
       #06FFS, #000BS, #000CS, #06FFS, #020ES, #020FS, #06FFS,
       #020ES, #020FS, #06FFS, #020ES, #020FS, #06FFS, #020ES,
```

```
        #020FS, #06FFS, #000BS, #000CS, #06FFS, #000BS, #000CS,
        #06FFS);

COMMENT  THE DPDA HAS 0016 REDUCE STATES;

ARRAY 0017 BYTE NUMTOPOP = (#00X, #01X, #00X, #01X, #01X, #02X, #00X,
        #00X, #02X, #02X, #00X, #02X, #02X, #00X, #00X, #00X, #02X);

ARRAY 0017 SHORT INTEGER REDUCESUCC = (#5830S, #0002S, #0003S,
        #0003S, #0500S, #0004S, #0205S, #0300S, #0300S, #0300S,
        #0502S, #0502S, #0502S, #0503S, #0503S, #0501S, #0700S);

COMMENT  THE DPDA HAS 0004 LOOK AHEAD STATES;

ARRAY 0004 BYTE LASYMNUM = (#00X, #00X, #00X, #00X);

ARRAY 0004 SHORT INTEGER SUCCSTATE = (#0206S, #0207S, #0208S, #0209S);

ARRAY 0004 SHORT INTEGER FAILSTATE = (#0007S, #0008S, #000DS, #000ES);

ARRAY 0008 BYTE LATABLE = (
        COMMENT  <EXPR>;
        #10X, #00X,

        COMMENT  <ARITH EXPR>;
        #16X, #00X,

        COMMENT  <ARITH EXPR>;
        #16X, #00X,

        COMMENT  <ARITH EXPR>;
        #16X, #00X);


COMMENT  THE DPDA HAS 0004 LOOKBACK STATES;

ARRAY 0004 BYTE LBSTART = (#00X, #02X, #05X, #08X);

ARRAY 0004 BYTE LBNUM = (#01X, #02X, #02X, #02X);

ARRAY 0011 SHORT INTEGER LBSTATE = (#0001S, #0000S, #0001S, #0003S,
        #0000S, #0006S, #0009S, #0000S, #000BS, #000CS, #0000S);

ARRAY 0011 SHORT INTEGER RESUMESTATE = (#0202S, #0203S, #0005S,
        #0005S, #020DS, #0301S, #0302S, #0303S, #020BS, #020CS,
        #020AS);
```

```
  COMMENT  THE SYMBOLS ACCESSING THE STATES;
  ARRAY 0015 BYTE SYMBEFOREREAD = (#00X, #01X, #0BX, #0CX, #0EX, #0FX,
         #04X, #11X, #12X, #05X, #06X, #07X, #08X, #12X, #12X);
  ARRAY 0004 BYTE SYMBEFORELA = (#11X, #12X, #12X, #12X);
  CLOSE BASE;
  ARRAY 18 INTEGER SEGTABLE = (
         #00000000,
         #00000088,
         #000000D8,
         #000000F6,
         #00000105,
         #0000012E,
         #0000017E,
         #00000190,
         #000001B2,
         #000001B6,
         #000001BE,
         #000001C6,
         #000001CE,
         #000001D2,
         #000001D6,
         #000001EC,
         #00000202,
         #00000211);
  COMMENT  END OF CARDS PUNCHED BY THE SLR(1) SYNTAX ANALYZER;

ALL DONE.
```

APPENDIX D

PROTO-COMPILER LISTING

```
$TITLE   PROTOCOM                                                              0001
 BEGIN                                                                         0002
 COMMENT  THIS PROGRAM IS A PROTO-COMPILER WRITTEN IN PL360                    0003
                                                                          ;    0004
  COMMENT  ASSUME THE FOLLOWING CARDS WERE PUNCHED BY THE                      0005
           SLR(1) SYNTAX ANALYZER (SLR1ANAL);                                  0006
  INTEGER NUMTERMINALS = 0010;                                                 0007
  INTEGER NUMNTS = 0009;                                                       0008
  INTEGER NUMSYMS = 0020;                                                      0009
                                                                               0010
                                                                               0011
  GLOBAL DATA SEGTABLE BASE R12;                                               0012
                                                                               0013
  ARRAY 0133 BYTE VSTRING = (#C5X, #D9X, #D9X, #D6X, #D9X, #E2X, #E8X,         0014
        #D4X, #40X, #6EX, #6DX, #4FX, #6DX, #4CX, #D7X, #D9X, #D6X,            0015
        #C7X, #D9X, #C1X, #D4X, #6EX, #4CX, #E2X, #E3X, #C1X, #E3X,            0016
        #C5X, #D4X, #C5X, #D5X, #E3X, #40X, #D3X, #C9X, #E2X, #E3X,            0017
        #6EX, #C5X, #D5X, #C4X, #4CX, #E2X, #E3X, #C1X, #E3X, #C5X,            0018
        #D4X, #C5X, #D5X, #E3X, #6EX, #4CX, #C1X, #E2X, #E2X, #C9X,            0019
        #C7X, #D5X, #D4X, #C5X, #D5X, #E3X, #6EX, #5EX, #4CX, #E5X,            0020
        #C1X, #D9X, #C9X, #C1X, #C2X, #D3X, #C5X, #6EX, #7EX, #4CX,            0021
        #C5X, #E7X, #D7X, #D9X, #6EX, #4CX, #C1X, #D9X, #C9X, #E3X,            0022
        #C8X, #40X, #C5X, #E7X, #D7X, #D9X, #6EX, #4CX, #E3X, #C5X,            0023
        #D9X, #D4X, #6EX, #4EX, #60X, #4CX, #D7X, #D9X, #C9X, #D4X,            0024
        #C1X, #D9X, #E8X, #6EX, #5CX, #61X, #4CX, #D5X, #E4X, #D4X,            0025
        #C2X, #C5X, #D9X, #6EX, #4CX, #C9X, #C4X, #C5X, #D5X, #E3X,            0026
        #C9X, #C6X, #C9X, #C5X, #D9X, #6EX);                                   0027
                                                                               0028
  ARRAY 0020 INTEGER LOCLENGTH = (#0000000A, #00000283, #00000983,             0029
        #00001001, #000012C1, #00001901, #00001941, #000018C1,                 0030
        #00001C01, #00001C48, #00001E4C, #00000349, #00000590,                 0031
        #00000A4B, #00000D0C, #0000104A, #00001306, #0000148C,                 0032
        #00001786, #00001989);                                                 0033
                                                                               0034
  COMMENT  THE DPDA HAS 0015 READ STATES;                                      0035
                                                                               0036
  ARRAY 0015 SHORT INTEGER READSTART = (#0000S, #0002S, #0004S, #0006S,        0037
        #0009S, #000BS, #000DS, #0010S, #0013S, #0016S, #0019S,                0038
        #001CS, #001FS, #0022S, #0025S);                                       0039
                                                                               0040
  ARRAY 0015 BYTE RDNUM = (#01X, #01X, #01X, #02X, #01X, #01X, #02X,           0041
```

```
        #02X, #02X, #02X, #02X, #02X, #02X, #02X, #02X);

ARRAY 0040 BYTE SYMLIST = (#01X, #00X, #0AX, #00X, #01X, #00X, #02X,
        #0AX, #00X, #03X, #00X, #04X, #00X, #09X, #0AX, #00X, #05X,
        #06X, #00X, #07X, #08X, #00X, #09X, #0AX, #00X, #09X, #0AX,
        #00X, #09X, #0AX, #00X, #09X, #0AX, #00X, #07X, #08X, #00X,
        #07X, #08X, #00X);

ARRAY 0040 SHORT INTEGER STATELIST = (#0001S, #06FFS, #020FS, #06FFS,
        #0210S, #06FFS, #0201S, #020FS, #06FFS, #0204S, #06FFS,
        #0006S, #06FFS, #020ES, #020FS, #06FFS, #0009S, #000AS,
        #06FFS, #000BS, #000CS, #06FFS, #020ES, #020FS, #06FFS,
        #020ES, #020FS, #06FFS, #020ES, #020FS, #06FFS, #020ES,
        #020FS, #06FFS, #000BS, #000CS, #06FFS, #000BS, #000CS,
        #06FFS);

COMMENT  THE DPDA HAS 0016 REDUCE STATES;

ARRAY 0017 BYTE NUMTOPOP = (#00X, #01X, #00X, #01X, #01X, #02X, #00X,
        #00X, #02X, #02X, #00X, #02X, #02X, #00X, #00X, #00X, #02X);

ARRAY 0017 SHORT INTEGER REDUCESUCC = (#0000S, #0002S, #0003S,
        #0003S, #0500S, #0004S, #0205S, #0300S, #0300S, #0300S,
        #0502S, #0502S, #0502S, #0503S, #0503S, #0501S, #0700S);

COMMENT  THE DPDA HAS 0004 LOOK AHEAD STATES;

ARRAY 0004 BYTE LASYMNUM = (#00X, #00X, #00X, #00X);

ARRAY 0004 SHORT INTEGER SUCCSTATE = (#0206S, #0207S, #0208S, #0209S);

ARRAY 0004 SHORT INTEGER FAILSTATE = (#0007S, #0008S, #000DS, #000ES);

ARRAY 0008 BYTE LATABLE = (
        COMMENT  <EXPR>;
        #10X, #00X,

        COMMENT  <ARITH EXPR>;
        #16X, #00X,

        COMMENT  <ARITH EXPR>;
        #16X, #00X,

        COMMENT  <ARITH EXPR>;
        #16X, #00X);
```

```
COMMENT  THE DPDA HAS 0004 LOOKBACK STATES;                                    0090
                                                                               0091
ARRAY 0004 BYTE LBSTART = (#00X, #02X, #05X, #08X);                            0092
                                                                               0093
ARRAY 0004 BYTE LBNUM = (#01X, #02X, #02X, #02X);                              0094
                                                                               0095
ARRAY 0011 SHORT INTEGER LBSTATE = (#0001S, #0000S, #0001S, #0003S,            0096
        #0000S, #0006S, #0009S, #0000S, #000BS, #000CS, #0000S);               0097
                                                                               0098
ARRAY 0011 SHORT INTEGER RESUMESTATE = (#0202S, #0203S, #0005S,                0099
        #0005S, #020DS, #0301S, #0302S, #0303S, #020BS, #020CS,                0100
        #020AS);                                                               0101
                                                                               0102
COMMENT  THE SYMBOLS ACCESSING THE STATES;                                     0103
                                                                               0104
ARRAY 0015 BYTE SYMBEFOREREAD = (#00X, #01X, #0BX, #0CX, #0EX, #0FX,           0105
        #04X, #11X, #12X, #05X, #06X, #07X, #08X, #12X, #12X);                 0106
                                                                               0107
ARRAY 0004 BYTE SYMBEFORELA = (#11X, #12X, #12X, #12X);                        0108
                                                                               0109
CLOSE BASE;                                                                    0110
                                                                               0111
ARRAY 18 INTEGER SEGTABLE = (                                                  0112
        #00000000,                                                             0113
        #00000088,                                                             0114
        #000000D8,                                                             0115
        #000000F6,                                                             0116
        #00000105,                                                             0117
        #0000012E,                                                             0118
        #0000017E,                                                             0119
        #00000190,                                                             0120
        #000001B2,                                                             0121
        #000001B6,                                                             0122
        #000001BE,                                                             0123
        #000001C6,                                                             0124
        #000001CE,                                                             0125
        #000001D2,                                                             0126
        #000001D6,                                                             0127
        #000001EC,                                                             0128
        #00000202,                                                             0129
        #00000211);                                                            0130
COMMENT  END OF CARDS PUNCHED BY THE SLR(1) SYNTAX ANALYZER;                   0131
                                                                               0132
INTEGER RESERVEDLIMIT = 0;                                                     0133
ARRAY 30 BYTE ALPHABET = ("ABCDEFGHIJKLMNOPQRSTUVWXYZ_$@#");                   0134
                                                                               0135
COMMENT  DECLARATIONS FOR THE SCANNER:                                         0136
                                                                               0137
```

```
          TOKEN IS THE INDEX INTO THE VOCABULARY V() OF THE                0138
          LAST SYMBOL SCANNED, CP IS THE POINTER TO THE LAST               0139
          CHARACTER SCANNED IN THE CARD IMAGE, CBCD IS THE                 0140
          LAST SYMBOL SCANNED;                                             0141
                                                                           0142
COMMENT   NUMBERVALUE CONTAINS THE NUMERIC VALUE OF THE LAST               0143
          CONSTANT SCANNED;                                                0144
INTEGER TOKEN = 1, PRODNUM, NUMBERVALUE, SP;                               0145
INTEGER REGISTER CP SYN R9;                                                0146
ARRAY 64 BYTE BCD;                                                         0147
ARRAY 3 BYTE EOFIL = (#6DX, #4FX,#6DX);                                    0148
ARRAY 12 BYTE IDENTS = ("<IDENTIFIER>");                                   0149
ARRAY 8 BYTE NUMB = (#4CX, #D5X, #E4X, #D4X, #C2X, #C5X, #D9X, #6EX);      0150
INTEGER DIVID = #00000061;                                                 0151
INTEGER LOCATION, LENGTH, VPT;                                             0152
ARRAY 80 BYTE CBUF;  COMMENT  CARD BUFFER;                                 0153
COMMENT  EXITFLAG IS USED TO INDICATE END OF COMPILING;                    0154
BYTE LISTFLAG, ENDIT, EXITFLAG = #00X;                                     0155
COMMENT  XR IS THE ERROR ROUTINE PARAMETER REGISTER;                       0156
INTEGER REGISTER XR SYN R5;                                                0157
SHORT INTEGER ERRCOUNT = 0S, CARDCOUNT = 0S;                               0158
INTEGER EOFILE = 0, NUMBER = 0,IDENT = 0, DIVIDE = 0;                      0159
LONG REAL CONWORK;  COMMENT  USED TO CONVERT TO DECIMAL;                   0160
BYTE TRUE = #FFX, FALSE = #00X;                                            0161
SHORT INTEGER PREVIOUSERROR, ERRLIMIT = 50S;                               0162
ARRAY 132 BYTE BLANK = 132(" ");                                           0163
ARRAY 132 BYTE WBUF;  COMMENT  WRITE BUFFER;                               0164
INTEGER MASK = #000000FF;                                                  0165
INTEGER MASK7 = #00000007;                                                 0166
INTEGER MASKFFFF = #0000FFFF;                                              0167
INTEGER BLANKMASK = #40404040;                                             0168
INTEGER MASK1 = #00000001;                                                 0169
INTEGER MASKFF00 = #0000FF00;                                              0170
LONG REAL VR3;                                                             0171
INTEGER VR3HI SYN VR3(0);                                                  0172
INTEGER VR3LOW SYN VR3(4);                                                 0173
ARRAY 256 BYTE CHARTYPE = 256(1);                                          0174
ARRAY 256 BYTE NOTLETTERORDIGIT = 256(1);                                  0175
ARRAY 256 INTEGER TX;                                                      0176
INTEGER TEXTLIMIT = 71;  COMMENT  SCAN TO CBUF(TEXTLIMIT);                 0177
ARRAY 10 BYTE NUMS = ("0123456789");                                       0178
BYTE PERIOD = #4BX;                                                        0179
ARRAY 8 BYTE CONBUF;                                                       0180
ARRAY 3 INTEGER TIME;                                                      0181
ARRAY 64 BYTE CBCD;                                                        0182
INTEGER SEGBASE, LATABSIZE, OFFSET;                                        0183
                                                                           0184
                                                                           0185
```

```
FUNCTION SD(10,#FB00);                                                  0186

FUNCTION SETZONE(8,#96F0);  COMMENT  FUNCTION TO SET ZONE;              0187
EXTERNAL PROCEDURE CLOSEM (R14);  NULL;                                 0188
                                                                        0189
COMMENT  DECLARE USEFUL LITERALS TO SIMPLIGY                            0190
         ACCESSING THE PARSING TABLES;                                  0191
                                                                        0192
   EQUATE AVSTRING        SYN 0;                                        0193
   EQUATE ALOCLENGTH      SYN 4;                                        0194
   EQUATE AREADSTART      SYN 8;                                        0195
   EQUATE ARONUM          SYN 12;                                       0196
   EQUATE ASYMLIST        SYN 16;                                       0197
   EQUATE ASTATELIST      SYN 20;                                       0198
   EQUATE ANUMTOPOP       SYN 24;                                       0199
   EQUATE AREDUCESUCC     SYN 28;                                       0200
   EQUATE ALASYMNUM       SYN 32;                                       0201
   EQUATE ASUCCSTATE      SYN 36;                                       0202
   EQUATE AFAILSTATE      SYN 40;                                       0203
   EQUATE ALATABLE        SYN 44;                                       0204
   EQUATE ALBSTART        SYN 48;                                       0205
   EQUATE ALBNUM          SYN 52;                                       0206
   EQUATE ALBSTATE        SYN 56;                                       0207
   EQUATE ARESUMSTATE     SYN 60;                                       0208
   EQUATE ASYMBEFOREREAD  SYN 64;                                       0209
   EQUATE BSYMBEFORELA    SYN 68;                                       0210
                                                                        0211
                                                                        0212
PROCEDURE FIND(R4);                                                     0213
   BEGIN ARRAY 4 INTEGER SAVEREGS;                                      0214
      STM(R1,R4,SAVEREGS);                                              0215
      MVC(63,BCD,BLANK);                                                0216
      R6 := R6 - R6; R1 := VPT SHLL 2;                                  0217
      R3 := ALOCLENGTH; R2 := SEGBASE + SEGTABLE(R3) + R1;              0218
      R3 := B2; R1 := R3 SHRL 6;                                        0219
      LOCATION := R1; R1 := R3 AND #3F;                                 0220
      LENGTH := R1;   R2 := LENGTH - 1;                                 0221
      FOR R1 := 0 STEP 1 UNTIL R2 DO                                    0222
      BEGIN                                                             0223
         R3 := LOCATION + R1; IC(R6,VSTRING(R3));                       0224
         STC(R6,BCD(R1));                                               0225
      END;                                                              0226
   LM(R1,R4,SAVEREGS);                                                  0227
   END;                                                                 0228
                                                                        0229
```

```
PROCEDURE ERROR(R4); COMMENT  PRINTS AND ACCOUNTS FOR ALL                0230
                              ERROR MESSAGES;                            0231
   BEGIN INTEGER SAVE4;                                                  0232
      SAVE4 := R4; R0 := ERRCOUNT + 1; ERRCOUNT := R0;                   0233
      IF R0 > ERRLIMIT THEN GOTO X;                                      0234
      COMMENT  IF LISTING IS SUPPRESSED, FORCE PRINTING OF               0235
               THE CARD BUFFER;                                          0236
      IF ¬LISTFLAG THEN                                                  0237
      BEGIN                                                              0238
         R0 := CARDCOUNT + 1; CVD(R0,CONWORK);                           0239
         UNPK(3,7,WBUF(45),CONWORK); SETZONE(WBUF(48));                  0240
         MVC(79,WBUF(52),CBUF); R0 := @WBUF; WRITE;                      0241
         MVC(131,WBUF,BLANK);                                            0242
      END;                                                               0243
      MVC(9,WBUF,"*** ERROR,");                                          0244
      CASE XR OF                                                         0245
      BEGIN                                                              0246
         NULL;  COMMENT  CASE 1 NOT USED;                                0247
         BEGIN                                                           0248
            R1 := @WBUF(CP+27); MVI("|",B1);                             0249
            MVC(17,WBUF(11),"ILLEGAL CHARACTER:");                       0250
         END;                                                            0251
         BEGIN                                                           0252
            MVC(15,WBUF(11),"STACK OVERFLOW. ");                         0253
            SET(EXITFLAG);                                               0254
         END;                                                            0255
         BEGIN                                                           0256
            R1 := @WBUF(CP+27); MVI("|",B1);                             0257
            MVC(19,WBUF(11),"ILLEGAL SYMBOL PAIR:");                     0258
         END;                                                            0259
         BEGIN                                                           0260
            MVC(24,WBUF(11),"PROGRAM ENDS PREMATURELY.");                0261
            SET(EXITFLAG);                                               0262
         END;                                                            0263
         BEGIN                                                           0264
            R1 := @WBUF(CP+27); MVI("|",B1);                             0265
            MVC(18,WBUF(11),"ILLEGAL IDENTIFIER:");                      0266
         END;                                                            0267
      END;                                                               0268
      R0 := @WBUF; WRITE; MVC(131,WBUF,BLANK);                           0269
      IF EXITFLAG THEN GOTO EXIT;                                        0270
      R0 := ERRCOUNT;                                                    0271
      IF R0 > 1 THEN                                                     0272
      BEGIN                                                              0273
         MVC(34,WBUF,"*** LAST ERROR DETECTED ON LINE  ");               0274
         R0 := PREVIOUSERROR; CVD(R0,CONWORK);                           0275
         UNPK(3,7,WBUF(33),CONWORK); SETZONE(WBUF(36));                  0276
         MVC(4,WBUF(37),". ***"); R0 := @WBUF; WRITE;                    0277
```

```
            MVC(41,WBUF,BLANK);                                          0278
         END;                                                            0279
         R0 := CARDCOUNT; PREVIOUSERROR := R0;                           0280
         R0 := ERRCOUNT;                                                 0281
         IF R0 = ERRLIMIT THEN                                           0282
         BEGIN                                                           0283
            MVC(20,WBUF,"*** TOO MANY ERRORS, ");                        0284
            MVC(21,WBUF(20)," CHECKING ABORTED. ***");                   0285
         END;                                                            0286
X:       R4 := SAVE4;                                                    0287
      END;                                                               0288
   PROCEDURE GETCARD(R4);                                                0289
   COMMENT  DOES ALL CARD READING AND LISTING;                           0290
      BEGIN INTEGER SAVE4;                                               0291
         SAVE4 := R4; R0 := @CBUF; READ;                                 0292
         IF ¬= THEN  COMMENT SIGNAL FOR EOF;                             0293
         BEGIN                                                           0294
            SET(ENDIT); MVC(79,CBUF,BLANK);                              0295
            MVC(10,CBUF,"EOF EOF EOF");                                  0296
         END;                                                            0297
         COMMENT  CARDCOUNT PRINTED ON LISTING;                          0298
         R0 := CARDCOUNT + 1; CARDCOUNT := R0;                           0299
         IF LISTFLAG THEN                                                0300
         BEGIN                                                           0301
            CVD(R0,CONWORK); UNPK(3,7,WBUF(20),CONWORK);                 0302
            SETZONE(WBUF(23)); MVC(79,WBUF(27),CBUF);                    0303
            R0 := @WBUF; WRITE; MVC(131,WBUF,BLANK);                     0304
         END;                                                            0305
         CP := 0; R4 := SAVE4;                                           0306
      END;                                                               0307
                                                                         0308
   PROCEDURE SCAN(R4);                                                   0309
      BEGIN INTEGER SAVE4, S1, S2;                                       0310
         SAVE4 := R4; R0 := 0; NUMBERVALUE := R0;                        0311
         WHILE TRUE DO                                                   0312
         BEGIN                                                           0313
            IF CP > TEXTLIMIT THEN GETCARD ELSE                          0314
            BEGIN  COMMENT  BRANCH ON NEXT CHARACTER IN CBUF;            0315
               IC(R1,CBUF(CP)); R1 := R1 AND MASK;                       0316
               IC(R2,CHARTYPE(R1)); R2 := R2 AND MASK;                   0317
               IF R2 < 10 THEN CASE R2 OF                                0318
               BEGIN                                                     0319
                  COMMENT  CASE 1: ILLEGAL CHARACTER;                    0320
                  BEGIN                                                  0321
                     IF ENDIT THEN                                       0322
                     BEGIN                                               0323
                        R1 := R1; TOKEN := R1; GOTO L1;                  0324
                     END;                                                0325
```

```
   XR := 2; ERROR;                                                    0326
END;                                                                  0327
COMMENT  CASE 2: BLANK;                                               0328
BEGIN  COMMENT  SKIP OVER BLANKS;                                     0329
   CP := CP + 1;                                                      0330
   IF CP <= TEXTLIMIT THEN                                            0331
   BEGIN                                                              0332
      IC(R2,CBUF(CP)); R2 := R2 AND MASK;                             0333
      WHILE R2 = 64 AND CP < TEXTLIMIT DO                             0334
      BEGIN                                                           0335
         CP := CP +1; IC(R2,CBUF(CP));                                0336
      END;                                                            0337
   END;                                                               0338
   IF CP ¬= TEXTLIMIT THEN CP := CP -1;                               0339
END;                                                                  0340
COMMENT  CASES 3 & 4 NOT USED;                                        0341
NULL;                                                                 0342
NULL;                                                                 0343
COMMENT  CASE 5: A LETTER;                                            0344
BEGIN                                                                 0345
   S1 := CP; R1 := 0; S2 := R1;                                       0346
   WHILE TRUE DO  COMMENT UNTIL END OF IDENT;                         0347
   BEGIN                                                              0348
      FOR CP := CP STEP 1 UNTIL TEXTLIMIT DO                          0349
      BEGIN                                                           0350
         IC(R1,CBUF(CP)); R1 := R1 AND MASK;                          0351
         IC(R2,NOTLETTERORDIGIT(R1));                                 0352
         R2 := R2 AND MASK;                                           0353
         IF R2 = 1 THEN                                               0354
         BEGIN  COMMENT  END OF IDENTIFIER;                           0355
            IF CP > S1 THEN R2 := CP - S1                             0356
            ELSE R2 := TEXTLIMIT - S1 + CP + 1;                       0357
            R4 := 0;                                                  0358
            FOR R1 := S1 STEP 1 UNTIL CP DO                           0359
            BEGIN                                                     0360
              IC(R3,CBUF(R1)); STC(R3,CBCD(R4));                      0361
              R4 := R4 + 1;                                           0362
            END;                                                      0363
         COMMENT  R2 IS LENGTH OF IDENT;                              0364
            R5 := NUMTERMINALS;                                       0365
            IF R2 <= RESERVEDLIMIT THEN                               0366
               FOR R1 := 1 STEP 1 UNTIL R5 DO                         0367
               BEGIN                                                  0368
                  VPT := R1; FIND; R3 := LENGTH - 1;                  0369
                  IF R2 = LENGTH THEN                                 0370
                  BEGIN                                               0371
                     R6 := @CBCD - 1; R8 := @BCD -1;                  0372
                     FOR R7 := 0 STEP 1 UNTIL R3 DO                   0373
```

```
                        BEGIN                                          0374
                          R6 := R6 + 1; R8 := R8 + 1;                  0375
                          CLC(0,B6,B8);                                0376
                          IF ¬= THEN GOTO OUT;                         0377
                        END;                                           0378
                        TOKEN := R1; GOTO L1;                          0379
                    OUT:                                               0380
                    END;                                               0381
                END;                                                   0382
            COMMENT  MUST BE <IDENT>;                                  0383
               R1 := IDENT;                                            0384
               IF R1 ¬= 0 THEN                                         0385
               BEGIN                                                   0386
                  TOKEN := R1; GOTO L1;                                0387
               END ELSE                                                0388
               BEGIN                                                   0389
                  XR := 6; ERROR; GOTO L2;                             0390
               END;                                                    0391
            END;                                                       0392
         END;                                                          0393
      COMMENT  END OF CARD;                                            0394
         GETCARD;                                                      0395
      END;                                                             0396
   END;                                                                0397
   COMMENT  CASE 6: DIGIT;                                             0398
   BEGIN                                                               0399
      R1 := NUMBER; TOKEN := R1; R1 := R1 - R1;                        0400
      WHILE TRUE DO  COMMENT  UNTIL GOTO L1;                           0401
      BEGIN                                                            0402
         FOR CP := CP STEP 1 UNTIL TEXTLIMIT DO                        0403
         BEGIN                                                         0404
            IC(R1,CBUF(CP));                                           0405
            IF R1 < 240 THEN GOTO L1;                                  0406
            R3 := NUMBERVALUE * 10 + R1 - 240;                         0407
         END;                                                          0408
         GETCARD;                                                      0409
      END;                                                             0410
   END;                                                                0411
   COMMENT  CASE 7: /;                                                 0412
   BEGIN                                                               0413
      R1 := DIVIDE; TOKEN := R1; CP := CP +1;                          0414
      GOTO L1;                                                         0415
   END;                                                                0416
   COMMENT  CASE 8: SPECIAL CHARACTER;                                 0417
   BEGIN                                                               0418
      R1 := R1 SHLL 2; R2 := TX(R1);                                   0419
      TOKEN := R2; CP := CP +1; GOTO L1;                               0420
   END;                                                                0421
```

```
                COMMENT  CASE 9: END OF FILE MARK,".";                    0422
                BEGIN                                                     0423
                   R1 := 1; TOKEN := R1; GOTO L1;                         0424
                END;                                                      0425
             END;  COMMENT  END OF CASE ON CHARTYPE;                      0426
             L2:                                                          0427
             CP := CP +1;                                                 0428
          END;                                                            0429
   END;                                                                   0430
L1:   R4 := SAVE4;                                                        0431
END;                                                                      0432
                                                                          0433
PROCEDURE PRINTIME(R6);                                                   0434
   BEGIN INTEGER SAVE6;                                                   0435
      SAVE6 := R6;                                                        0436
      UNPK(7,3,CONBUF,TIME(0));                                           0437
      MVC(1,WBUF(18),CONBUF(1));  MVC(0,WBUF(20),":");                    0438
      MVC(1,WBUF(21),CONBUF(3));                                          0439
      R6 := SAVE6;                                                        0440
   END;                                                                   0441
PROCEDURE PRINTDATE(R6);                                                  0442
   BEGIN INTEGER SAVE6, SAVE15;                                           0443
      SAVE6 := R6; SAVE15 := R15;                                         0444
      R1 := 2; SVC(11);                                                   0445
      R15 := SAVE15; R0 := R0 OR  #F;                                     0446
      TIME(0) := R0;                                                      0447
      TIME(4) := R1; UNPK(7,3,CONBUF,TIME(4));                            0448
      MVC(1,WBUF(9),CONBUF(3));                                           0449
      MVC(0,WBUF(11),PERIOD); MVC(2,WBUF(12),CONBUF(5));                  0450
      R6 := SAVE6;                                                        0451
   END;                                                                   0452
                                                                          0453
PROCEDURE INITIALIZE(R4);                                                 0454
COMMENT  THIS PROCEDURE SETS VARIABLES TO BE USED IN THE                  0455
         PROGRAM.  IT SHOULD ALSO SAVE THE CURRENT TIME OF                0456
         DAY TO BE USED IN PROCEDURE SUMMARIZE;                           0457
   BEGIN INTEGER SAVE4;                                                   0458
      SAVE4 := R4; MVC(131,WBUF,BLANK); R0 := @WBUF;                      0459
      MVC(34,WBUF,"   REPLACE THIS HEADING WITH ONE OF");                 0460
      MVC(36,WBUF(36),"YOUR OWN -- VERSION OF DECEMBER 1972.");           0461
      WRITE; MVC(72,WBUF,BLANK); WRITE; WRITE; PRINTDATE;                 0462
      PRINTIME; MVC(8,WBUF,"TODAY IS ");                                  0463
      MVC(0,WBUF(16),"@");                                                0464
      R0 := @WBUF;                                                        0465
      WRITE; MVC(131,WBUF,BLANK); WRITE; WRITE;                           0466
      SEGBASE := R12;  COMMENT  SAVE BEGINNING ADDRESS OF                 0467
                                PARSING TABLES;                           0468
      R1 := NUMTERMINALS + 1 SHRL 3;                                      0469
```

```
R2 := NUMTERMINALS + 1 AND #7;                                          0470
IF R2 > 0 THEN R1 := R1 + 1;                                            0471
LATABSIZE := R1;                                                        0472
COMMENT  SEARCH THE TERMINAL SYMBOLS FOR "_|_",                         0473
         "<NUMBER>", "<IDENT>", AND "/";                                0474
                                                                        0475
R5 := NUMTERMINALS;                                                     0476
FOR R1 := 1 STEP 1 UNTIL R5 DO                                          0477
BEGIN                                                                   0478
R4 := R4 - R4; R7 := R7 - R7; R3 := R3 - R3;                            0479
   VPT := R1; FIND; R2 := LENGTH;                                       0480
   IF R2 > RESERVEDLIMIT THEN RESERVEDLIMIT := R2;                      0481
   IF R2 = 1 THEN                                                       0482
   BEGIN                                                                0483
      IC(R3,BCD(0));                                                    0484
      IF R3 < #C0 THEN                                                  0485
      BEGIN                                                             0486
         R4 := @CHARTYPE(R3); MVI(8,B4);                                0487
         R3 := R3 SHLL 2; TX(R3) := R1;                                 0488
      END;                                                              0489
      IC(R3,BCD(0)); R3 := R3 AND MASK;                                 0490
      IF R3 = DIVID THEN DIVIDE := R1;                                  0491
   END ELSE                                                             0492
   IF R2 = 3 THEN                                                       0493
   BEGIN                                                                0494
      FOR R3 := 0 STEP 1 UNTIL 2 DO                                     0495
      BEGIN                                                             0496
         IC(R4,BCD(R3)); IC(R7,EOFIL(R3));                              0497
         R4 := R4 AND MASK; R7 := R7 AND MASK;                          0498
         IF R4 ¬= R7 THEN GOTO YYY;                                     0499
      END;                                                              0500
      EOFILE := R1;                                                     0501
   END ELSE                                                             0502
   IF R2 = 12 THEN                                                      0503
   BEGIN                                                                0504
      FOR R3 := 0 STEP 1 UNTIL 10 DO                                    0505
      BEGIN                                                             0506
         IC(R4,BCD(R3)); IC(R7,IDENTS(R3));                             0507
         R4 := R4 AND MASK; R7 := R7 AND MASK;                          0508
         IF R4 ¬= R7 THEN GOTO YYY;                                     0509
      END;                                                              0510
      IDENT := R1;                                                      0511
   END ELSE                                                             0512
   IF R2 = 8 THEN                                                       0513
   BEGIN                                                                0514
      FOR R3 := 0 STEP 1 UNTIL 7 DO                                     0515
      BEGIN                                                             0516
         IC(R4,BCD(R3)); IC(R7,NUMB(R3));                               0517
```

```
                R4 := R4 AND MASK; R7 := R7 AND MASK;                                  0518
                IF R4 ¬= R7 THEN GOTO YYY;                                             0519
             END;                                                                      0520
             NUMBER := R1;                                                             0521
          END;                                                                         0522
       YYY:                                                                            0523
       END;                                                                            0524
       R5 := EOFILE;                                                                   0525
       R5 := R5 SHLL 2; R1 := LOCLENGTH(R5) SHRL 6; R2 := @VSTRING(R1);                0526
       MVC(2,B2,"EOF");                                                                0527
       R3 := R3 - R3; IC(R3," "); R4 := @CHARTYPE(R3);                                 0528
       MVI(2,B4);                                                                      0529
                                                                                       0530
       R3 := R3 - R3;                                                                  0531
       FOR R1 := 0 STEP 1 UNTIL 9 DO                                                   0532
       BEGIN                                                                           0533
          IC(R3,NUMS(R1)); R4 := @CHARTYPE(R3); MVI(6,B4);                             0534
          R4 := @NOTLETTERORDIGIT(R3); MVI(0,B4);                                      0535
       END;                                                                            0536
       COMMENT  LENGTH OF ALPHABET IS 30;                                              0537
       FOR R1 := 0 STEP 1 UNTIL 29 DO                                                  0538
       BEGIN                                                                           0539
          R3 := R3 - R3; IC(R3,ALPHABET(R1));                                          0540
          R4 := @CHARTYPE(R3); MVI(5,B4);                                              0541
          R4 := @NOTLETTERORDIGIT(R3); MVI(0,B4);                                      0542
          R3 := R3 SHLL 2; TX(R3) := R1;                                               0543
       END;                                                                            0544
       CP := TEXTLIMIT + 1; MVC(79,CBUF,BLANK); R1 := 0;                               0545
       SP := R1; SET(LISTFLAG); RESET(ENDIT); R4 := SAVE4;                             0546
    END;  COMMENT  END INITIALIZE;                                                     0547
                                                                                       0548
PROCEDURE EMIT(R4); NULL;                                                              0549
COMMENT  THIS PROCEDURE HAS THE RESPONSIBILITY OF SETTING                              0550
         THE NEXT ELEMENT OF THE CODE ARRAY TO THE OPCODE                              0551
         DETERMINED BY PROCEDURE SYNTHESIZE;                                           0552
                                                                                       0553
PROCEDURE LOOKUP(R4); NULL;                                                            0554
COMMENT  THIS PROCEDURE LOOKS UP A NAME IN THE SYMBOL TABLE                            0555
         AND ENTERS IT IF NOT THERE;                                                   0556
                                                                                       0557
PROCEDURE SYNTHESIZE(R4); NULL;                                                        0558
COMMENT THIS PROCEDURE IS RESPONSIBLE FOR THE SEMANTICS OF                             0559
         THE COMPILER.  IT TAKES THE FORM OF A LARGE CASE                              0560
         STATEMENT ON GLOBAL VARIABLE PRODNUM.  ARRIVE HERE                            0561
         FROM THE CASE STATEMENT IN PROCEDURE ANALYZE;                                 0562
                                                                                       0563
PROCEDURE PRINTSUMMARY(R4);                                                            0564
COMMENT  THIS PROCEDURE SHOULD SUBTRACT CURRENT TIME OF DAY                            0565
```

```
          WITH THAT SAVED IN PROCEDURE INITIALIZE AND STORE                0566
          THE RESULT IN WBUF STARTING  IN COLUMN 19;                       0567
      BEGIN INTEGER SAVE4;                                                 0568
          INTEGER SAVE15;                                                  0569
          SAVE15 := R15;                                                   0570
          SAVE4 := R4;                                                     0571
         R1 := 2; SVC(11);                                                 0572
          R15 := SAVE15;                                                   0573
          R0 := R0 OR #F; TIME(8) := R0;                                   0574
          SD(3,3,TIME(8),TIME(0));                                         0575
          UNPK(7,3,CONBUF,TIME(8)); MVC(1,WBUF(21),CONBUF(3));             0576
          MVC(0,WBUF(23),PERIOD); MVC(1,WBUF(24),CONBUF(5));               0577
          MVC(6,WBUF(28),"SECONDS");                                       0578
          MVC(17,WBUF,"TIME IN EXECUTION:"); R0 := @WBUF; WRITE;           0579
          MVC(40,WBUF,BLANK);                                              0580
          R4 := SAVE4;                                                     0581
      END;                                                                 0582
                                                                           0583
  GLOBAL PROCEDURE ANALYZE(R4);                                            0584
      BEGIN INTEGER SAVE4, NEXTSYMBOL;                                     0585
          ARRAY 150 INTEGER STATESTACK = 150(0);                           0586
          INTEGER STATENUM =0, LASYMBOL = 0;                               0587
                                                                           0588
          PROCEDURE PUSHANDREAD(R4);                                       0589
             BEGIN INTEGER SAVE4;                                          0590
                SAVE4 := R4; R1 := SP;                                     0591
                IF R1 < 150 THEN                                           0592
                BEGIN                                                      0593
                   R1 := R1 + 1; SP := R1;                                 0594
                END ELSE                                                   0595
                BEGIN                                                      0596
                   XR := 3; SET(EXITFLAG); ERROR;                          0597
                END;                                                       0598
                R1 := TOKEN; NEXTSYMBOL := R1;                             0599
                COMMENT  SET VAR(SP) TO BCD AND VAL(SP) TO                 0600
                            NUMBERVALUE;                                   0601
                SCAN; R4 := SAVE4;                                         0602
             END;  COMMENT  END PUSHANDREAD;                               0603
                                                                           0604
          INTEGER CYCLECNT = 0; SAVE4 := R4;                               0605
          WHILE TRUE DO                                                    0606
          BEGIN                                                            0607
             R1 := CYCLECNT + 1; CYCLECNT := R1;                           0608
             R1 := SP SHLL 2; R2 := STATESTACK(R1) SHRL 8 + 1;             0609
             CASE R2 OF                                                    0610
             BEGIN                                                         0611
                COMMENT  CASE 1, READ VIA LINEAR SEARCH;                   0612
                BEGIN                                                      0613
```

```
   PUSHANDREAD; R1 := STATENUM SHLL 1;                                  0614
   R2 := AREADSTART;                                                    0615
   R4 := SEGBASE + SEGTABLE(R2) + R1; LH(R2,B4);                        0616
   R1 := R1 SHRL 1; R5 := ARDNUM;                                       0617
   R4 := SEGBASE + SEGTABLE(R5) + R1;                                   0618
   IC(R3,B4); R3 := R3 AND MASK;                                        0619
   R3 := R3 + R2; R4 := NEXTSYMBOL;                                     0620
   R5 := ASYMLIST;                                                      0621
   R6 := SEGBASE + SEGTABLE(R5) + R2; IC(R5,B6);                        0622
   R5 := R5 AND MASK;                                                   0623
   WHILE R4 ¬= R5 AND R2 < R3 DO                                        0624
   BEGIN                                                                0625
      R2 := R2 + 1; R6 := R6 + 1; IC(R5,B6);                            0626
   END;                                                                 0627
   R2 := R2 SHLL 1;                                                     0628
   R4 := ASTATELIST;                                                    0629
   R3 := SEGBASE + SEGTABLE(R4) + R2;                                   0630
   LH(R1,B3); R1 := R1 AND MASKFFFF;                                    0631
   R2 := SP SHLL 2; STATESTACK(R2) := R1;                               0632
   R1 := R1 AND MASK; STATENUM := R1;                                   0633
END;                                                                    0634
COMMENT  CASE 2, READ VIA AN ARRAY ACCESSS;                             0635
BEGIN                                                                   0636
   PUSHANDREAD; R1 := STATENUM SHLL 1;                                  0637
   R2 := AREADSTART;                                                    0638
   R3 := SEGBASE + SEGTABLE(R2) + R1; LH(R2,B3);                        0639
   R2 := R2 + NEXTSYMBOL SHLL 1; R1 := ASTATELIST;                      0640
   R3 := SEGBASE + SEGTABLE(R1) + R2; LH(R1,B3);                        0641
   R1 := R1 AND MASKFFFF;                                               0642
   R2 := SP SHLL 2; STATESTACK(R2) := R1;                               0643
   R1 := R1 AND MASK; STATENUM := R1;                                   0644
END;                                                                    0645
COMMENT  CASE 3, REDUCE;                                                0646
BEGIN                                                                   0647
   R1 := STATENUM; PRODNUM := R1; SYNTHESIZE;                           0648
   R1 := STATENUM; R2 := ANUMTOPOP;                                     0649
   R3 := SEGBASE + SEGTABLE(R2) + R1;                                   0650
   IC(R2,B3);                                                           0651
   R2 := R2 AND MASK; R3 := SP - R2; SP := R3;                          0652
   R1 := R1 SHLL 1; R2 := AREDUCESUCC;                                  0653
   R3 := SEGBASE + SEGTABLE(R2) + R1; LH(R2,B3);                        0654
   R1 := SP SHLL 2; STATESTACK(R1) := R2;                               0655
   R2 := R2 AND MASK; STATENUM := R2;                                   0656
END;                                                                    0657
COMMENT  CASE 4, LOOK AHEAD (ORDINARY);                                 0658
BEGIN                                                                   0659
   R1 := TOKEN; LASYMBOL := R1; R2 := R1 SHRL 3;                        0660
   R5 := STATENUM * LATABSIZE + R2;                                     0661
```

```
      R3 := R1 AND MASK7; R4 := 7 - R3;                                  0662
      R3 := ALATABLE;                                                    0663
      R6 := SEGBASE + SEGTABLE(R3) + R5; IC(R1,B6);                      0664
      R1 := R1 AND MASK SHRL R4 AND MASK1;                               0665
      IF R1 = 1 THEN                                                     0666
      BEGIN                                                              0667
         R1 := STATENUM SHLL 1;                                          0668
         R2 := ASUCCSTATE;                                               0669
         R3 := SEGBASE + SEGTABLE(R2) + R1;                              0670
         LH(R2,B3);                                                      0671
      END ELSE                                                           0672
      BEGIN                                                              0673
         R1 := STATENUM SHLL 1;                                          0674
         R2 := AFAILSTATE;                                               0675
         R3 := SEGBASE + SEGTABLE(R2) + R1;                              0676
         LH(R2,B3);                                                      0677
      END;                                                               0678
      R2 := R2 AND MASKFFFF; R1 := SP SHLL 2;                            0679
      STATESTACK(R1) := R2; R2 := R2 AND MASK;                           0680
      STATENUM := R2;                                                    0681
   END;                                                                  0682
   COMMENT  CASE 5, LOOK AHEAD (FOR A PRODECTION                         0683
            WITH AN EMPTY RIGHT PART);                                   0684
   BEGIN                                                                 0685
      R1 := TOKEN; LASYMBOL := R1; R2 := R1 SHRL 3;                      0686
      R5 := STATENUM * LATABSIZE + R2;                                   0687
      R3 := R1 AND MASK7; R4 := 7 - R3;                                  0688
      R3 := ALATABLE;                                                    0689
      R4 := SEGBASE + SEGTABLE(R3) + R5; IC(R1,B4);                      0690
      R1 := R1 AND MASK SHRL R4 AND MASK1;                               0691
      IF R1 = 1 THEN                                                     0692
      BEGIN                                                              0693
         R1 := SP SHLL 2;                                                0694
         R2 := STATESTACK(R1) SHRL 8;                                    0695
         WHILE R2 = 3 OR R2 = 4 DO                                       0696
         BEGIN                                                           0697
            R3 := STATESTACK(R1) AND MASK SHLL 1;                        0698
            R4 := AFAILSTATE;                                            0699
            R5 := SEGBASE + SEGTABLE(R4) + R3;                           0700
            LH(R4,B5);                                                   0701
            STATESTACK(R1) := R4;                                        0702
            R2 := R4 SHRL 8;                                             0703
         END;                                                            0704
         R1 := SP + 1; SP := R1;                                         0705
         R1 := STATENUM SHLL 1;                                          0706
         R2 := ASUCCSTATE;                                               0707
         R3 := SEGBASE + SEGTABLE(R2) + R1; LH(R2,B3);                   0708
      END ELSE                                                           0709
```

```
   BEGIN                                                                0710
      R1 := STATENUM SHLL 1;                                            0711
      R2 := AFAILSTATE;                                                 0712
      R3 := SEGBASE + SEGTABLE(R2) + R1; LH(R2,B3);                     0713
   END;                                                                 0714
   R1 := SP SHLL 2; STATESTACK(R1) := R2;                               0715
   R2 := R2 AND MASK; STATENUM := R2;                                   0716
END;                                                                    0717
COMMENT  CASE 6, LOOK BACK;                                             0718
BEGIN                                                                   0719
   R1 := SP - 1 SHLL 2; R2 := STATENUM;                                 0720
   R3 := ALBSTART;                                                      0721
   R4 := SEGBASE + SEGTABLE(R3) + R2; IC(R3,B4);                        0722
   R3 := R3 AND MASK;                                                   0723
   R4 := ALBNUM;                                                        0724
   R5 := SEGBASE + SEGTABLE(R4) + R2; IC(R4,B5);                        0725
   R4 := R4 AND MASK + R3;                                              0726
   R3 := R3 SHLL 1; R5 := ALBSTATE;                                     0727
   R7 := SEGBASE + SEGTABLE(R5) + R3;                                   0728
   LH(R5,B7);                                                           0729
   R6 := STATESTACK(R1); R3 := R3 SHRL 1;                               0730
   WHILE R6 ¬= R5 AND R3 < R4 DO                                        0731
   BEGIN                                                                0732
      R3 := R3 + 1 SHLL 1; R5 := ALBSTATE;                              0733
   R7 := SEGBASE + SEGTABLE(R5) + R3; LH(R5,B7);                        0734
      R3 := R3 SHRL 1;                                                  0735
   END;                                                                 0736
   R3 := R3 SHLL 1; R1 := ARESUMSTATE;                                  0737
   R7 := SEGBASE + SEGTABLE(R1) + R3; LH(R1,B7);                        0738
   R2 := SP SHLL 2; STATESTACK(R2) := R1;                               0739
   R1 := R1 AND MASK; STATENUM := R1;                                   0740
END;                                                                    0741
COMMENT  CASE 7, ERROR;                                                 0742
BEGIN INTEGER PREVERRCYCLE = #FFFFFFFF;                                 0743
   R0 := @WBUF; WRITE;                                                  0744
   R1 := CYCLECNT - 2; CYCLECNT := R1;                                  0745
   IF R1 ¬= PREVERRCYCLE THEN                                           0746
   BEGIN                                                                0747
      PREVERRCYCLE := R1; R1 := SP - 1 SHLL 2;                          0748
      R2 := STATESTACK(R1);                                             0749
      IF R2 < 512 THEN                                                  0750
      BEGIN                                                             0751
         R2 := R2 AND MASK;                                             0752
         R3 := ASYMBEFOREREAD;                                          0753
         R4 := SEGBASE + SEGTABLE(R3) + R2;                             0754
         IC(R3,B4);                                                     0755
      END ELSE                                                          0756
      BEGIN                                                             0757
```

```
      R2 := STATENUM; R3 := BSYMBEFORELA;                          0758
      R4 := SEGBASE + SEGTABLE(R3) + R2; IC(R3,B4);                0759
    END;                                                           0760
    R3 := R3 AND MASK; R2 := 33;                                   0761
    FOR R7 := 0 STEP 1 UNTIL 1 DO                                  0762
    BEGIN                                                          0763
      VPT := R3; FIND; R1 := LENGTH - 1;                           0764
      FOR R6 := 0 STEP 1 UNTIL R1 DO                               0765
      BEGIN                                                        0766
        IC(R5,BCD(R6)); STC(R5,WBUF(R2));                          0767
        R2 := R2 + 1;                                              0768
      END;                                                         0769
      R2 := R2 + 2; R3 := STATENUM;                                0770
      IF R3 = 255 THEN R3 := NEXTSYMBOL                            0771
      ELSE R3 := LASYMBOL;                                         0772
      R3 := R3 AND MASK;                                           0773
    END;                                                           0774
    XR := 4; ERROR;                                                0775
    MVC(131,WBUF,BLANK);                                           0776
    MVC(17,WBUF,"PARTIAL PARSE IS: ");                             0777
    R0 := @WBUF; WRITE; MVC(17,WBUF,BLANK);                        0778
    R2 := SP - 1 SHLL 2;                                           0779
    FOR R1 := 8 STEP 4 UNTIL R2 DO                                 0780
    BEGIN                                                          0781
       R3 := STATESTACK(R1);                                       0782
       IF R3 < 512 THEN                                            0783
       BEGIN                                                       0784
          R3 := R3 AND MASK;                                       0785
          R4 := ASYMBEFOREREAD;                                    0786
          R5 := SEGBASE+SEGTABLE(R4)+R3;                           0787
          IC(R4,B5);                                               0788
       END ELSE                                                    0789
       BEGIN                                                       0790
          R3 := R3 AND MASK;                                       0791
          R4 := BSYMBEFORELA;                                      0792
          R5 := SEGBASE+SEGTABLE(R4)+R3;                           0793
          IC(R4,B5);                                               0794
       END;                                                        0795
       R4 := R4 AND MASK; VPT := R4; FIND;                         0796
       MVC(63,WBUF(4),BCD); WRITE;                                 0797
       MVC(63,WBUF(4),BLANK);                                      0798
    END;                                                           0799
END;                                                               0800
R1 := NEXTSYMBOL;                                                  0801
IF R1 = 1 THEN                                                     0802
BEGIN                                                              0803
    XR := 5; ERROR;                                                0804
END ELSE                                                           0805
```

```
                    BEGIN                                                    0806
                       VPT := R1; FIND; R1 := LENGTH - 1;                    0807
                       MVC(16,WBUF,"THE INPUT SYMBOL,");                     0808
                       MVC(63,WBUF(20),BCD);                                 0809
                       MVC(16,WBUF(60)," WILL BE IGNORED:");                 0810
                       R0 := @WBUF;                                          0811
                       WRITE; MVC(131,WBUF,BLANK);                           0812
                    END;                                                     0813
                    R1 := STATENUM;                                          0814
                    IF R1 = 255 THEN                                         0815
                    BEGIN                                                    0816
                    COMMENT  ERROR OCCURRED IN A READ STATE;                 0817
                       R1 := SP - 1; SP := R1; R1 := R1 SHLL 2;              0818
                       R2 := STATESTACK(R1) AND MASK;                        0819
                       STATENUM := R2;                                       0820
                    END ELSE                                                 0821
                    BEGIN                                                    0822
                    COMMENT  ERROR OCCURRED IN A LOOK-AHEAD STATE;           0823
                       SCAN;  COMMENT  SKIP THE NEXT SYMBOL;                 0824
                       R1 := R1 SHLL 1;                                      0825
                    R2 := ASUCCSTATE;                                        0826
                    R4 := SEGBASE + SEGTABLE(R2) + R1;                       0827
                       LH(R2,B4); R2 := R2 AND MASKFF00;                     0828
                       LH(R3,B4); R3 := R3 AND MASK;                         0829
                       R4 := ANUMTOPOP; R5 := SEGBASE+SEGTABLE(R4)+R3;       0830
                       IC(R4,B5); R4 := R4 AND MASK;                         0831
                       IF R2 = 512 AND R4 ¬= 255 THEN                        0832
                          R1 := STATENUM OR #00000300                        0833
                       ELSE R1 := STATENUM OR #00000400;                     0834
                       R2 := SP SHLL 2; STATESTACK(R2) := R1;                0835
                    END;                                                     0836
                 END;  COMMENT  END OF CASE 7;                               0837
                 BEGIN  COMMENT EXIT;                                        0838
                    R1 := 1; TOKEN := R1; R1 := 0;                           0839
                    SP := R1; STATESTACK(R1) := R1;                          0840
                    STATENUM := R1; GOTO XXX;                                0841
                 END;                                                        0842
              END;  COMMENT  END OF CASE(STATETYPE);                         0843
           END;                                                              0844
XXX:       R4 := SAVE4;                                                      0845
        END;                                                                 0846
                                                                             0847
PROCEDURE MAIN(R4);                                                          0848
   BEGIN INTEGER SAVE4;                                                      0849
        SAVE4 := R4;                                                         0850
        INITIALIZE;                                                          0851
        ANALYZE;                                                             0852
        PRINTSUMMARY;                                                        0853
```

```
      R4 := SAVE4;                                                          0854
   END;                                                                     0855
                                                                            0856
MAIN;                                                                       0857
EXIT:                                                                       0858
MVC(17,WBUF,"END OF COMPILATION"); R0 := @WBUF; WRITE;                      0859
END.                                                                        0860
```

APPENDIX E

PROTO-COMPILER OUTPUT

```
   REPLACE THIS HEADING WITH ONE OF YOUR OWN -- VERSION OF DECEMBER 1972.

TODAY IS 72.339 @ 17:34

                    0001        XRAY = 1; YELLOW = 1  XRAY*XRAY;
                                                      |

*** ERROR, ILLEGAL SYMBOL PAIR:  <TERM>  <IDENTIFIER>
PARTIAL PARSE IS:
    <STATEMENT LIST>
    <VARIABLE>
    =
    <TERM>
THE INPUT SYMBOL,   <IDENTIFIER>                              WILL BE IGNORED:

                    0002        ZEBRA = 2/1 + YELLOW;    ABLE = ZEBRA *1+ZEBRA-
                    0003      BETA = ABLE / XRAY + ZEBRA / XRAY;  END
                    0004   EOF EOF EOF
TIME IN EXECUTION:   00.02  SECONDS
END OF COMPILATION
```

```
*********************************************************
*          OS/360 OPERATING SYSTEM INTERFACE FOR PL360
*********************************************************

          ICTL     1,71,18
          SPACE
*********************************************************
*********************************************************
          SPACE
          MACRO
&EP       ENTER
          ENTRY    &EP
          USING    &EP,15
&EP       STM      12,2,SAVE                SAVE REGISTERS
          L        12,=A($PL360IO)          ESTABLISH ADDRESSING
          USING    $PL360IO,12
          DROP     15
          MEND
          SPACE
          MACRO
          EXIT
          LM       12,2,SAVE                RESTORE REGISTERS
          BR       14
          DROP     12
          MEND
          SPACE    2
$PL360IO  CSECT
          SPACE
LINELEN   EQU      132                      PRINTER LINE LENGTH
LINESMAX  EQU      60                       PRINTER LINES/PAGE
          PRINT    NOGEN
          SPACE
*    GLOBAL PROCEDURE READ(R14)
*         (R0)  = BUFFER ADDRESS
*         (R13) = SAVE AREA ADDRESS
*         (R14) = RETURN ADDRESS
READ      ENTER
          TM       SYSIN+DOPEN,OPENMASK     TEST FOR OPEN DCB
          BO       READ1
          LR       2,0
          OPEN     (SYSIN,(INPUT))          ISSUE OPEN
          LR       0,2
READ1     CLI      EOF,X'FF'                TEST FOR PREVIOUS END
          BE       ERRPROC
          GET      SYSIN,(0)                GET CARD
READ2     CLI      EOF,0                    SET CONDITION CODE
          EXIT
          SPACE
*                    EOD EXIT ROUTINE
          USING    $PL360IO,12
ENDRDR    MVI      EOF,X'FF'                EODAD EXIT
          B        READ2
          DROP     12
          SPACE    2
*    GLOBAL PROCEDURE WRITE(R14)
*         (R0)  = BUFFER ADDRESS
*         (R13) = SAVE AREA ADDRESS
*         (R14) = RETURN ADDRESS
WRITE     ENTER
          LR       2,0
          TM       SYSOUT+DOPEN,OPENMASK TEST FOR OPEN DATA SE
          BO       WRITE1
          OPEN     (SYSOUT,(OUTPUT))
WRITE1    PUT      SYSOUT,(1)               GET NEXT BUFFER ADDRE
          MVC      0(1,1),CARRCONT          SUPPLY CONTROL CHARAC
          CLI      CARRCONT,C'1'
          BNE      WRITE2
          MVI      LINECNT,0                RESET LINE COUNT AND
          MVI      CARRCONT,C' '
```

```
WRITE2   MVC     1(LINELEN,1),0(2)      TRANSFER BUFFER
         IC      2,LINECNT              INCREMENT LINE COUNT
         LA      2,1(,2)
         STC     2,LINECNT
         CLI     LINECNT,LINESMAX       TEST FOR FULL PAGE
         BL      WRITE3
         MVI     CARRCONT,C'1'          SET SKIP
WRITE3   LM      12,2,SAVE
         BR      14
         DROP    12
         SPACE   2
*    GLOBAL PROCEDURE PAGE(R14)
*         (R14) = RETURN ADDRESS
         ENTRY   PAGE
         USING   PAGE,15
PAGE     MVI     CARRCONT,C'1'          SET
         BR      14
         SPACE   2
*    GLOBAL PROCEDURE PUNCH(R14)
*         (R0)  = BUFFER ADDRESS
*         (R13) = SAVE AREA ADDRESS
*         (R14) = RETURN ADDRESS
PUNCH    ENTER
         TM      SYSPUNCH+DOPEN,OPENMASK
         BO      PUNCH1
         LR      2,0
         OPEN    (SYSPUNCH,(OUTPUT))
         LR      0,2
PUNCH1   PUT     SYSPUNCH,(0)           PUT CARD IMAGE
         EXIT
         SPACE   2
SYSOUT   DCB     DSORG=PS,MACRF=PL,DDNAME=SYSPRINT,DEVD=DA,R
                 LRECL=LINELEN+1,BFTEK=S,EROPT=ABE
SYSIN    DCB     DSORG=PS,MACRF=GM,DDNAME=SYSIN,DEVD=DA,RECF
                 LRECL=80,BFTEK=S,EROPT=ABE,EODAD=ENDRDR
SYSPUNCH DCB     DSORG=PS,MACRF=PM,DDNAME=SYSPUNCH,DEVD=DA,R
                 LRECL=80,BFTEK=S,EROPT=ABE
SAVE     DS      7F
ERRPROC  DC      H'0'
EOF      DC      X'00'
LINECNT  DC      X'00'                      PRINTER LINE COUN
CARRCONT DC      C'1'                   PRINTER CARRIAGE CONT
         SPACE
         DCBD
$PL360IO CSECT
DOPEN    EQU     DCBOFLGS-IHADCB        BYTE SET BY OPEN
OPENMASK EQU     X'10'
         END
```

APPENDIX G

JOB CONTROL LANGUAGE

```
*************************************
* JCL TO EXECUTE THE SYNTAX ANALYZER *
*************************************

//W0000564    JOB (0564,0812NT,CS14),'WOODS'
//JOBLIB      DD DSN=C0012,PL360,DISP=OLD,VOL=SER=MARY,UNIT=2314
//COMP        EXEC PGM=PL360,REGION=150K
//SYSPRINT    DD SYSOUT=A,SPACE=TRK,(10,5)),
//            DCB=(RECFM=FBA,LRECL=133,BLKSIZE=798)
//SYSGO       DD UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),DISP=(,PASS),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD *

        COMMENT  THE SYNTAX ANALYZER;

//LINK        EXEC PGM=IEWL,PARM='MAP,LIST',REGION=100K,
            COND=(0,NE,COMP)
//SYSUT1      DD UNIT=SYSDA,SPACE=(TRK,(20,10))
//SYSLMOD     DD DSN=&T(PROGRAM),UNIT=SYSDA,DISP=(,PASS),
//            SPACE=(CYL,(1,1,1),RLSE)
//SYSLIB      DD DSN=SD938.PLIO,UNIT=2314,VOL=SER=MARY,DISP=SHR
//SYSPRINT    DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210),
//            SPACE=(TRK,5)
//SYSLIN      DD DSN=*.COMP.SYSGO,DISP=(OLD,DELETE)
//            DD *
//GO          EXEC PGM=*.LINK.SYSLMOD,REGION=100K,COND=(4,LT,LINK)
//SYSPRINT    DD SYSOUT=A,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            SPACE=(TRK,(10,5))
//SYSPUNCH    DD SYSOUT=B,DCB=BLKSIZE=800
//SYSIN       DD *
DATA CARDS.
/*
```

```
************************************
* JCL TO EXECUTE THE PROTO-COMPILER  *
************************************

//WOOO564   JOB (0564,0812NT,CS14),'WOODS'
//JOBLIB    DD DSN=CJOJ2,PL360,DISP=OLD,VOL=SER=MARY,UNIT=2314
//COMP      EXEC PGM=PL360,REGION=100K
//SYSPRINT  DD SYSOUT=A,SPACE=TRK,(10,5)),
//          DCB=(RECFM=FBA,LRECL=133,BLKSIZE=798)
//SYSGO     DD UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),DISP=(,PASS),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN     DD *

       COMMENT  THE PROTO-COMPILER

//LINK      EXEC PGM=IEWL,PARM='MAP,LIST',REGION=100K,
          COND=(0,NE,COMP)
//SYSUT1    DD UNIT=SYSDA,SPACE=(TRK,(20,10))
//SYSLMOD   DD DSN=&T(PROGRAM),UNIT=SYSDA,DISP=(,PASS),
//          SPACE=(CYL,(1,1,1),RLSE)
//SYSLIB    DD DSN=SO938.PLIO,UNIT=2314,VOL=SER=MARY,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210),
//          SPACE=(TRK,5)
//SYSLIN    DD DSN=*.COMP.SYSGO,DISP=(OLD,DELETE)
//          DD *
//GO        EXEC PGM=*.LINK.SYSLMOD,REGION=60K,COND=(4,LT,LINK)
//SYSPRINT  DD SYSOUT=A,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//          SPACE=(TRK,(10,5))
//SYSIN     DD *
DATA CARDS.
/*
```

LIST OF REFERENCES

1. Wirth, N., "PL360, A Programming Language for the 360 Computers," J. ACM, v. 15, p. 37-74, 1968.

2. Blanchard, R. C., Steps Toward a PL360-Based Compiler Generator for the IBM 360 Computer, M. S. Thesis, Naval Postgraduate School, Monterey, California, June 1972.

3. DeRemer, F. L., "Simple LR(k) Grammars," Comm. ACM, v. 14, p. 453-460, 1971.

4. McKeeman, W. M., Horning, J. J., and Wortman, D. B., A Compiler Generator, Prentice-Hall, Englewood Cliffs, N. J., 1970.

5. Feldman, J., and Gries, D., "Translator Writing Systems," Comm. ACM, v. 11, p. 77-113, 1968.

6. Wirth, N., and Weber, H., "EULER - A generalization of ALGOL, and its Formal Definition," Comm. ACM, v. 9, p. 13-25, p. 89-99, 1966.

7. OS/360 PL360 Compiler, IBM Contributed Library (Type IV) Program, Number 360D-03.2.011.

8. Malcolm, M. A., PL360 (Revised), A Programming Language for the IBM 360, Stanford Univ., May 1971.

9. CEP Rep. 2, Simple LR(k) Grammars: Definition and Implementation, by R. L. DeRemer, U. of Calif., Santa Cruz, 4 September 1970.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Instructor R. H. Brubaker, Code 53Bh Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
4.	Asst. Professor G. E. Heidorn, Code 55Hd Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
5.	Asst. Professor G. A. Kildall, Code 53Kd Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
6.	LT R. A. Woods, USN 212 North Park Drive Hutchinson, Kansas 67501	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3 REPORT TITLE
A PL360-Based Compiler Generating System

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Master's Thesis; December 1972

5 AUTHOR(S) (First name, middle initial, last name)
Robert Allen Woods

6. REPORT DATE
December 1972

7a. TOTAL NO. OF PAGES
112

7b. NO. OF REFS
9

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

A compiler generating system written in the language PL360 to run on IBM System/360 computers is presented. The concepts and principles of the XPL compiler generating system are reviewed. The SLR(k) parsing algorithm is briefly described, and an example of SLR(1) parsing is presented. A description of the compiler generating system is presented along with its limitations, and instructions for its use are given. The required PL360 to System/360 interface is described and a listing is included. Program listings and sample input and output are included in the appendices.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Compiler Generator						
PL360						
Proto-Compiler						
Syntax Analyzer						
Syntax Checker						
SLR(k) Parser						

141857

Thesis
W8435 Woods
c.1 A PL360-based compiler
generating system.